آموزش جامع C#

www.tahlildadeh.com

مولف:مهندس افشین رفوآ

بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

کتاب آموزشی برنامه نویسی با C#

نویسنده مهندس افشین رفوآ

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

تقدیم به نائب امام عصر، آیت الله خامنه ای

که عصا زدنش ضرب آهنگ حیدری دارد.

تقدیم به همه جویندگان علم که توان و امکان شرکت در کلاس های حضوری ما را ندارند.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

فهرست

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

مقدمه

زکات علم نشر آن است. حضرت علی(ع)

موسسه آموزشی تحلیل داده ، با حضور جمعی از متخصصین مجرب در زمینه برنامه نویسی در نظر دارد،مطالب آموزشی خود را در قالب کتاب های آموزشی و فیلم ، به صورت رایگان در دسترس عموم قرار دهد تا حتی آن دسته از عزیزانی که بنا به دلایل مالی،مسافت جغرافیایی و یا نداشتن وقت کافی ، امکان شرکت در دوره های حضوری برای آنها میسر نیست،از یادگیری بی بهره نمانند.

علاوه بر این علاقه مندان می توانند ، با ثبت نام در انجمن سایت تحلیل داده،سوالات خود را مطرح نموده و مدرسین آموزشگاه و اعضای انجمن در اسرع وقت،پاسخ های خود را، حتی الامکان به صورت فیلم، در دسترس عموم قرار دهند.

لذا از کلیه فعالان در این زمینه دعوت می شود، در این حرکت جمعی در کنار ما باشند و با حضور فعال خود در انجمن،گام موثری در بهبود سطح علمی جوانان کشور عزیزمان،ایران بردارند.

آدرس سایت : http://www.tahlildadeh.com

توجه :

برای دانلود سورس کد مثال های کتاب ، اینجا را کلیک کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

مقدمه ای بر MS Visual Studio

C#، که سی شارپ تلفظ می شود، یک زبان برنامه نویسی برای دستور دادن به کامپیوتر می باشد. این دستورات را می توان از طریق برنامه ی ساده ای مثل NotePad هم نوشت. شیوه ی دیگر برای نوشتن دستور، استفاده از محیط برنامه نویسی می باشد که با مجهز بودن به ابزار گوناگون عملیات گوناگونی همچون کار با پروژه ها، ایجاد فایل های ضروری و توزیع برنامه ی کاربردی تکمیل شده را بسیار سهل و آسان می کند.

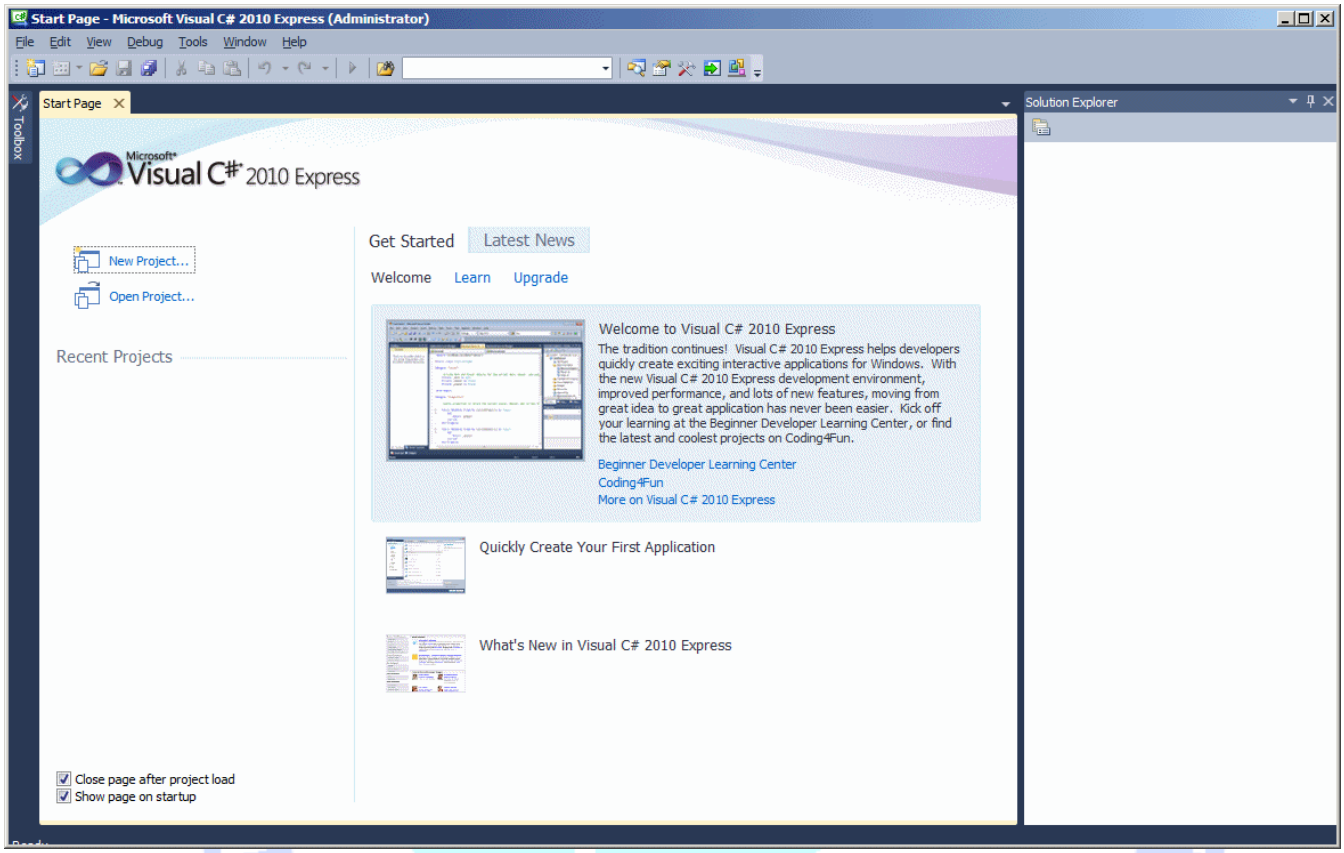
محبوب ترین محیط برنامه نویسی برای نوشتن دستورات C#، Microsoft Visual Studio می باشد. چنانچه، هدف اصلی شما یادگیری این زبان برنامه نویسی است، می توانید از Microsoft Visual C# 2010 Express کمک بگیرید.

برای یادگیری این دروس، می توانید از Microsoft Visual Studio 2010 یا Microsoft Visual C# 2010 Express استفاده کنید. برای دانلود Microsoft Visual C# 2010 Express، کافی است به سایت مایکروسافت مراجعه کرده و روی لینک Visual Studio Express کلیک کنید.

راه اندازی Microsoft Visual C# Express

به منظور راه اندازی برنامه ی Visual C# 2010 Express، روی گزینه های زیر کلیک کنید : Start -> (All) Programs -> Microsoft Visual Studio 2010 Express -> Microsoft Visual C# 2010 Express

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



رابط (کاربری) The Microsoft Visual Studio

فهرست گزینه ی اصلی (main menu)

بالاترین قسمت رابط برنامه ی Microsoft Visual Studio منوی اصلی را نشان می دهد که به دسته هایی همچون File و Edit تقسیم شده. منو اصلی همان کارایی فهرست گزینه ی یک برنامه ی کاربردی ساده را دارد.

نوارابزارها (Toolbars)

زیر (قسمت) منوی اصلی بخشی است به نام نوارابزار (toolbar). معمولاً، هنگامی که برنامه بالا می آید، نوارابزار استاندارد (Standard toolbar) را به عنوان پیش فرض خود نمایش می دهد. تعدادی نوارابزار اضافه برسازمان هم در نتیجه ی گزینه هایی که شما انتخاب می کنید یا عملیاتی که انجام می دهید نمای

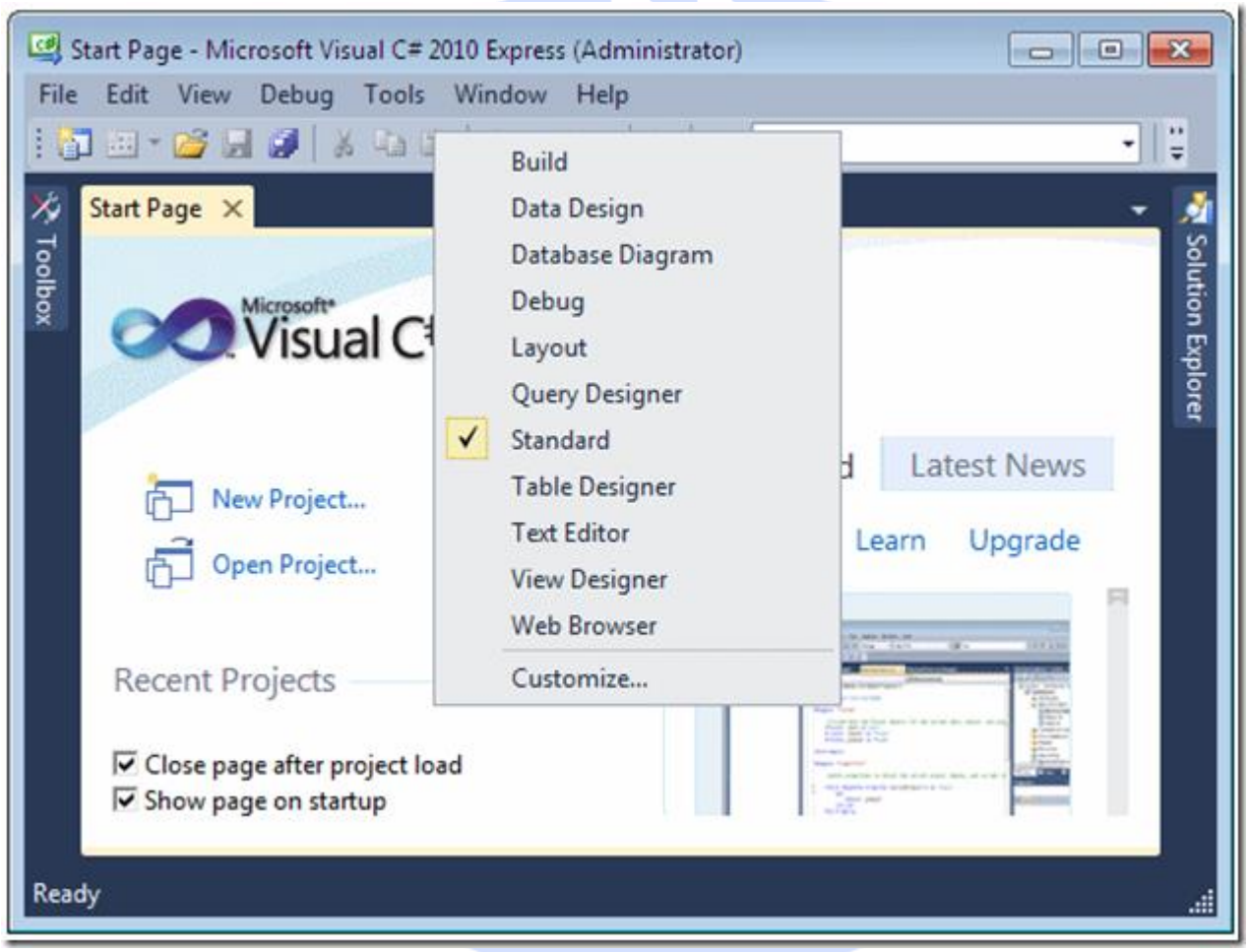
ش داده می شود. برای انجام این کار (دیدن نوارابزارهای فرعی)

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

در بخش main menu روی View -> Toolbars کلیک کرده، سپس toolbar مورد نظر را انتخاب کنید.

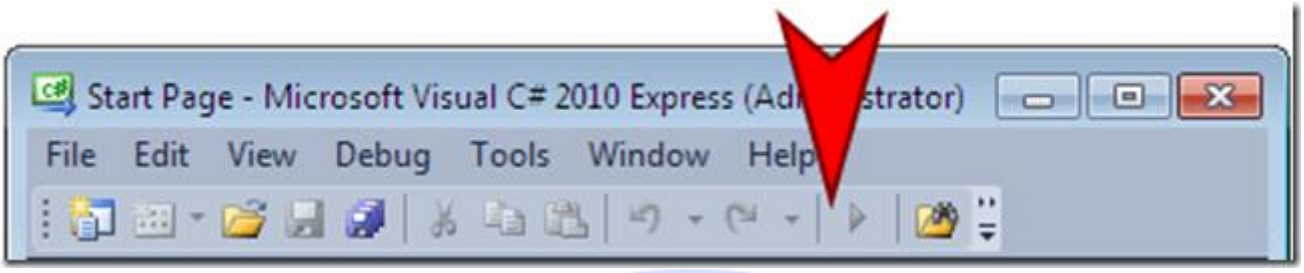
روی main menu یا هر toolbar

دلخواه دیگر راست کلیک کنید. فهرستی نمایان می شود که می توان در آن نوارابزار دلخواه را انتخاب کرد.



فهرست ها و نوارابزارهای Microsoft Visual Studio را می توان مطابق میل تنظیم (customize) کرد. می توان با اضافه کردن یک menu item به فهرست گزینه، آن را مطابق میل تنظیم کرد یا با اضافه کردن تنها یک دکمه به نوارابزار، آن را (سفارشی) تغییر داد. برای شروع، روی هر گزینه ای (در main menu یا toolbar) راست کلیک کرده، سپس گزینه ی customize را انتخاب کنید. می خواهیم آیتمی به نام Start without debugging را به سمت چپ Start button اضافه کنیم.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



برای این کار مراحل زیر را دنبال کنید.

روی گزینه ی customize (در main menu یا toolbar) کلیک کنید.

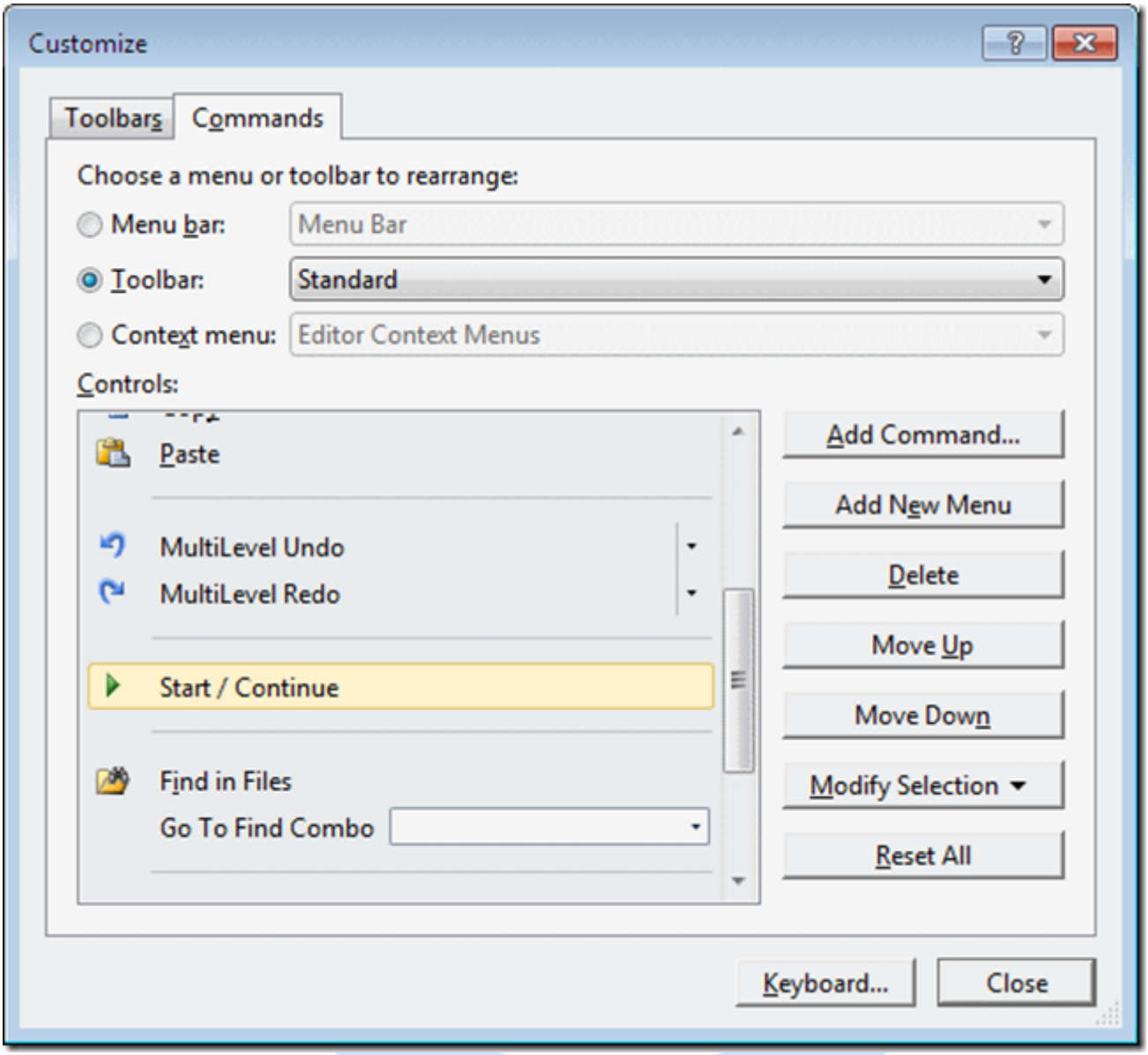
در پنجره ی محاوره (dialog box)، روی تَب Commands کلیک کنید.

روی Toolbars radio button کلیک کنید.

در قسمت Toolbars combo box، آن نوارابزاری که میزبان دکمه ی مورد نظر خواهد بود را انتخاب کنید. برای مثال، نوارابزار Standard.

در لیست Controls، روی دکمه ی بعدی کلیک کنید. در این مثال، Start / Continue

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

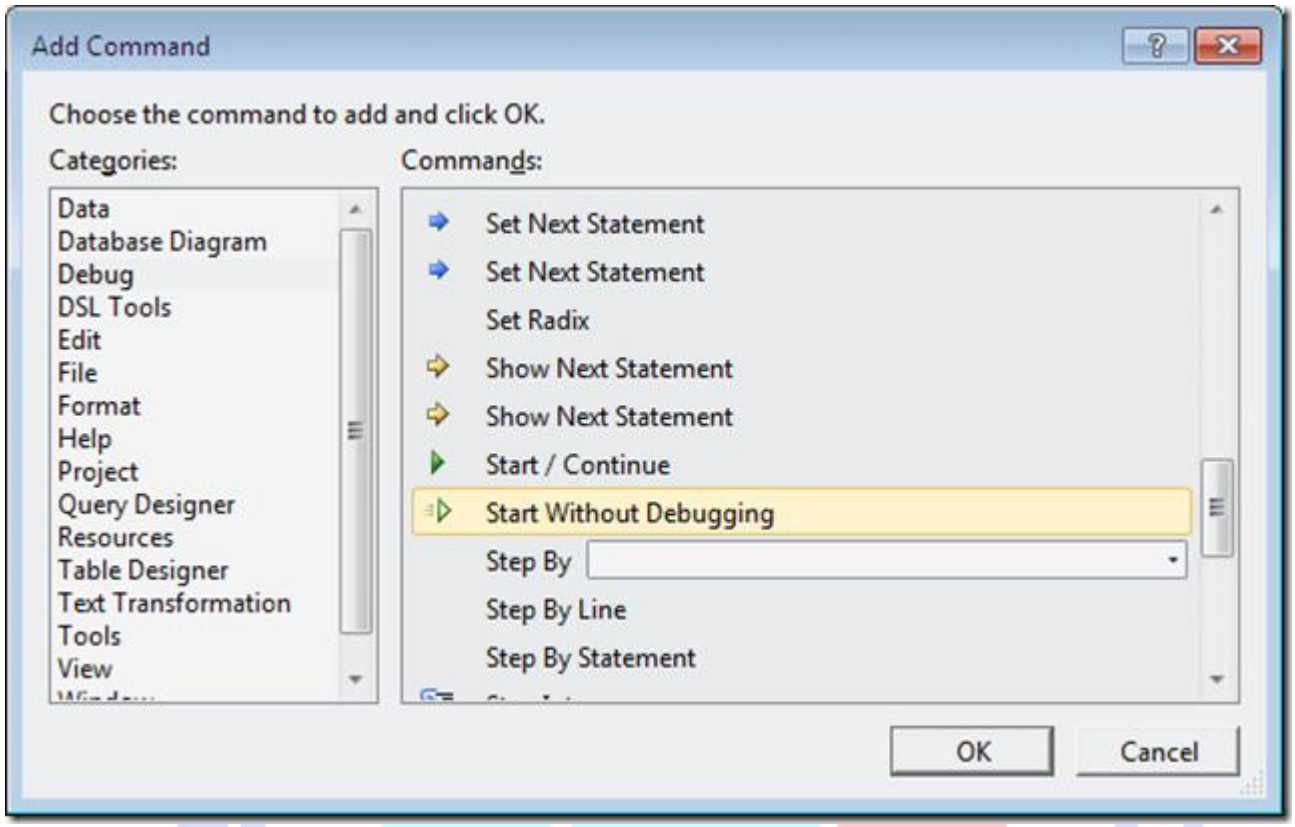


روی Add Command کلیک کنید.

در لیست Categories، دسته ای را انتخاب کنید که دکمه ی موردنظرشما را دارد. برای مثال، Debug

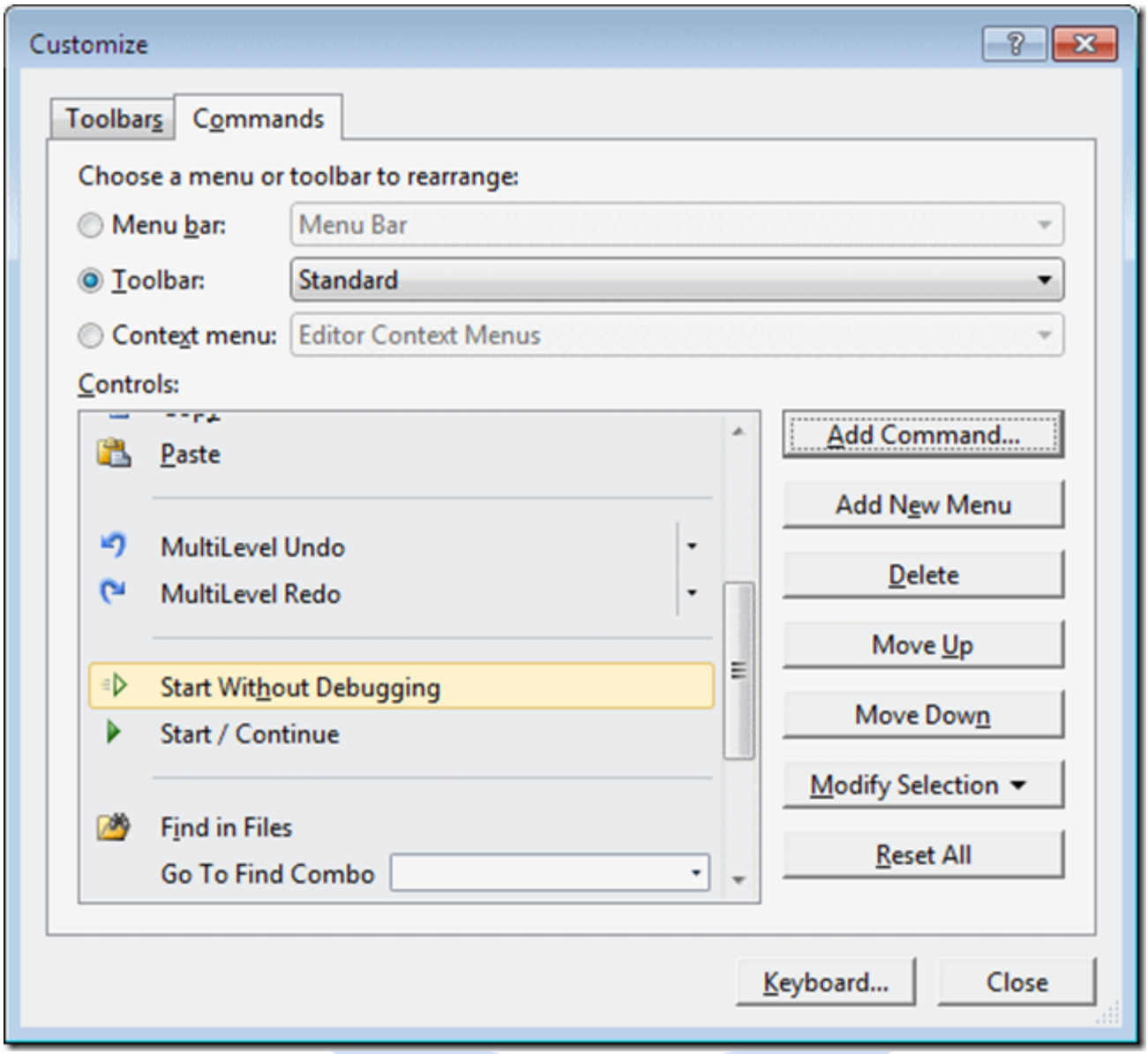
حال، آیتم منو یا دکمه ی مورد نظر را از لیست Commands انتخاب کنید. در این مثال، منظور Start Without Debugging می باشد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



سپس، روی گزینه ی ok کلیک کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



اکنون می توانید روی گزینه ی close کلیک کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

صفحه شروع (the Start page)

Start page، بدنه ی اصلی برنامه ی Microsoft Visual C# Express/Microsoft Visual Studio محسوب می شود. هنگام بالا آمدن برنامه، تبی را (در بالاترین قسمت) مشاهده می کنید که Start page نام دارد. بخش سمت چپ برنامه دو برچسب دارد : به نام های New Project... و Open Project... آن دسته از پروژه هایی که قبلاً به وجود آمده و مورد استفاده قرار گرفته اند، تحت Open Project به نمایش گذاشته می شوند. هنگامی که در حال کار با پروژه ی خاصی هستید، Start page در پس زمینه فعال است و توسط یک تب نشان داده می شود. برای مشاهده ی Start page

روی برچسب Start page کلیک کنید.

سپس، (در main menu)، گزینه ی View -> Start Page را انتخاب کنید.

نحوه ی بازکردن و بستن پنجره

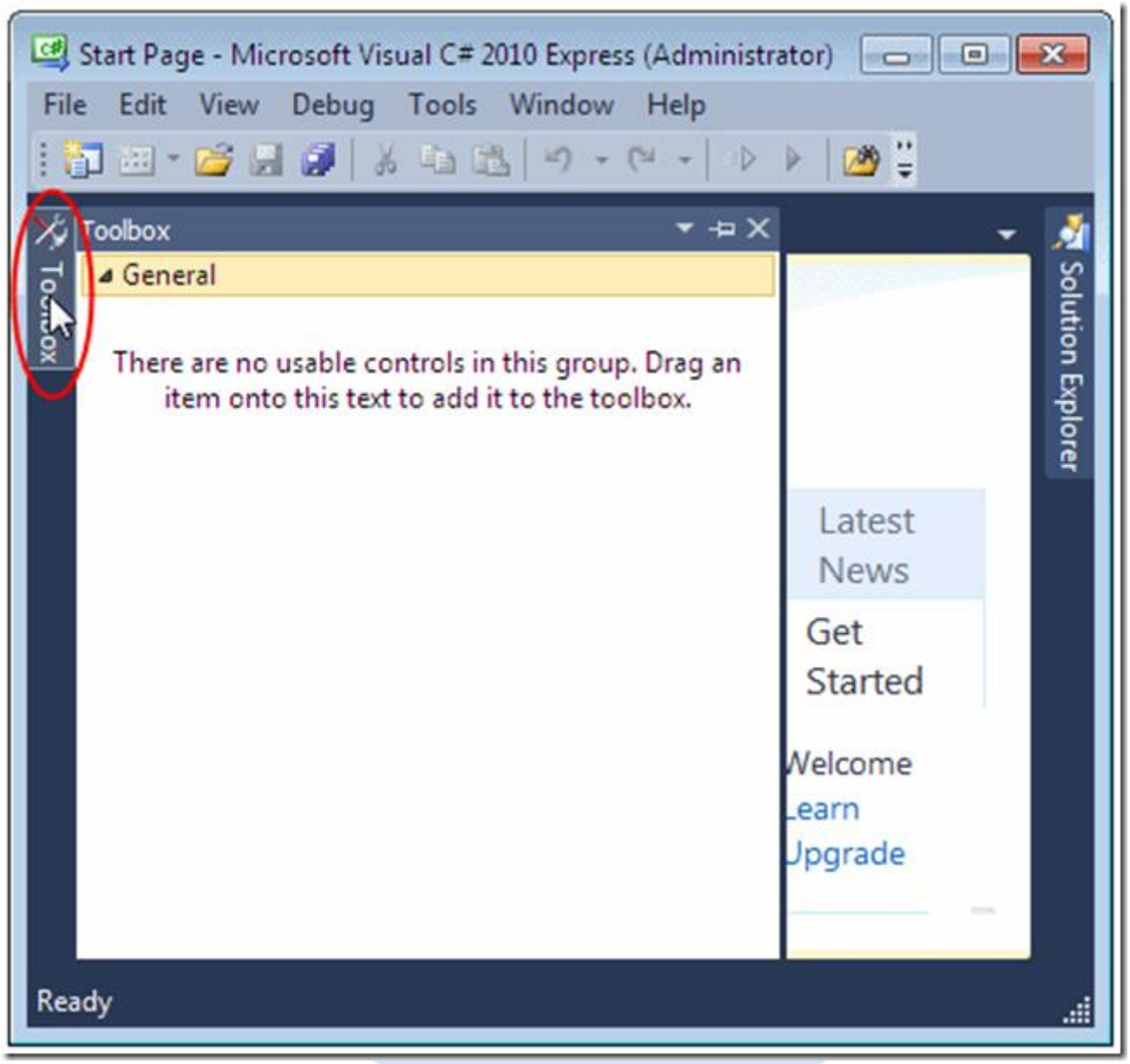
پس از بالا آمدن برنامه، تعدادی پنجره در اختیار شما قرار می گیرد. پنجره های مزبور مرتب مورد استفاده ی کاربران قرار می گیرند. چنانچه پنجره ای مناسب کار شما نیست، می توانید به راحتی آن را از صفحه حذف کنید. برای بستن پنجره نیز همان طور که می دانید باید روی گزینه ی close آن کلیک کنید.

تمامی پنجره های انتخابی در فهرست View Menu به نمایش گذاشته شده. بنابراین، می توانید پنجره ی دلخواه را از فهرست نام برده انتخاب کنید.

پنهان سازی پنجره به صورت خودکار (auto-hide)

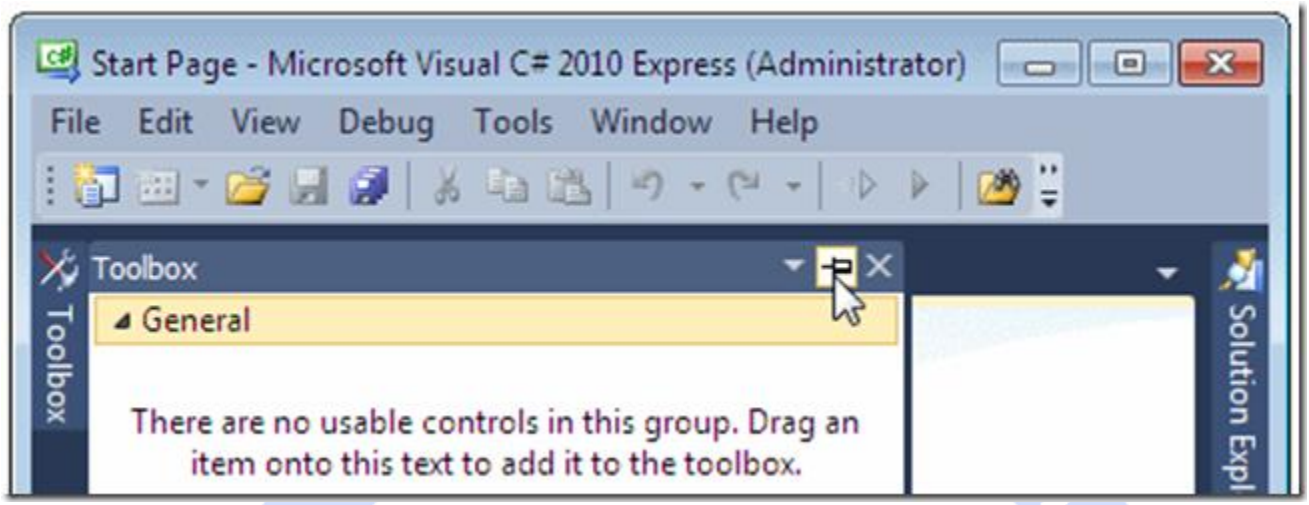
هنگام کار با پروژه، از پنجره هایی استفاده می کنیم که هریک کاربردی جداگانه دارند. برخی از پنجره ها با تنها یک آیکون نشان داده می شوند و بدنه ی اصلی آن ها نمایش داده نمی شود. به منظور مشاهده چنین پنجره ای، کافی است مکان نمای موس را روی آن قرار دهید. این کار پنجره ی مذکور را باز می کند.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

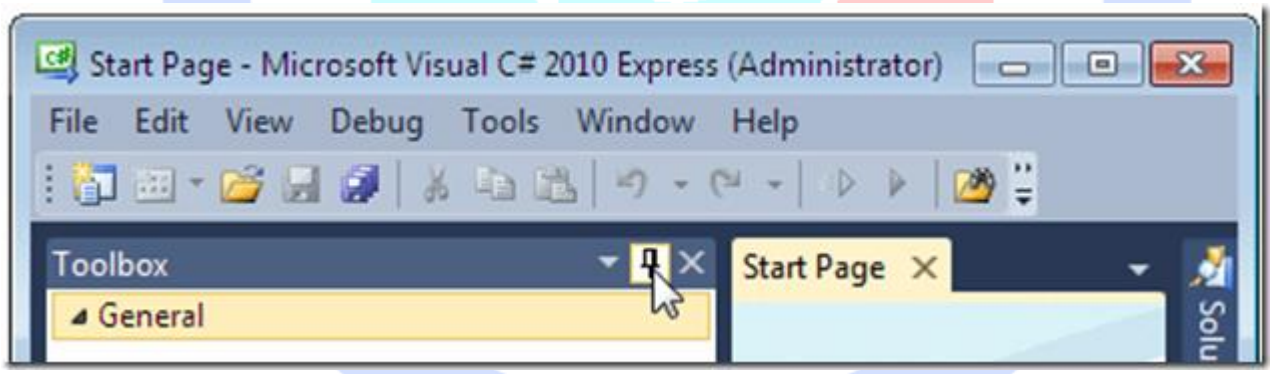


پس از باز کردن پنجره، نوارعنوان (title bar) ای با سه دکمه پدیدار می شود. یکی از دکمه های نام برده Auto hide می باشد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



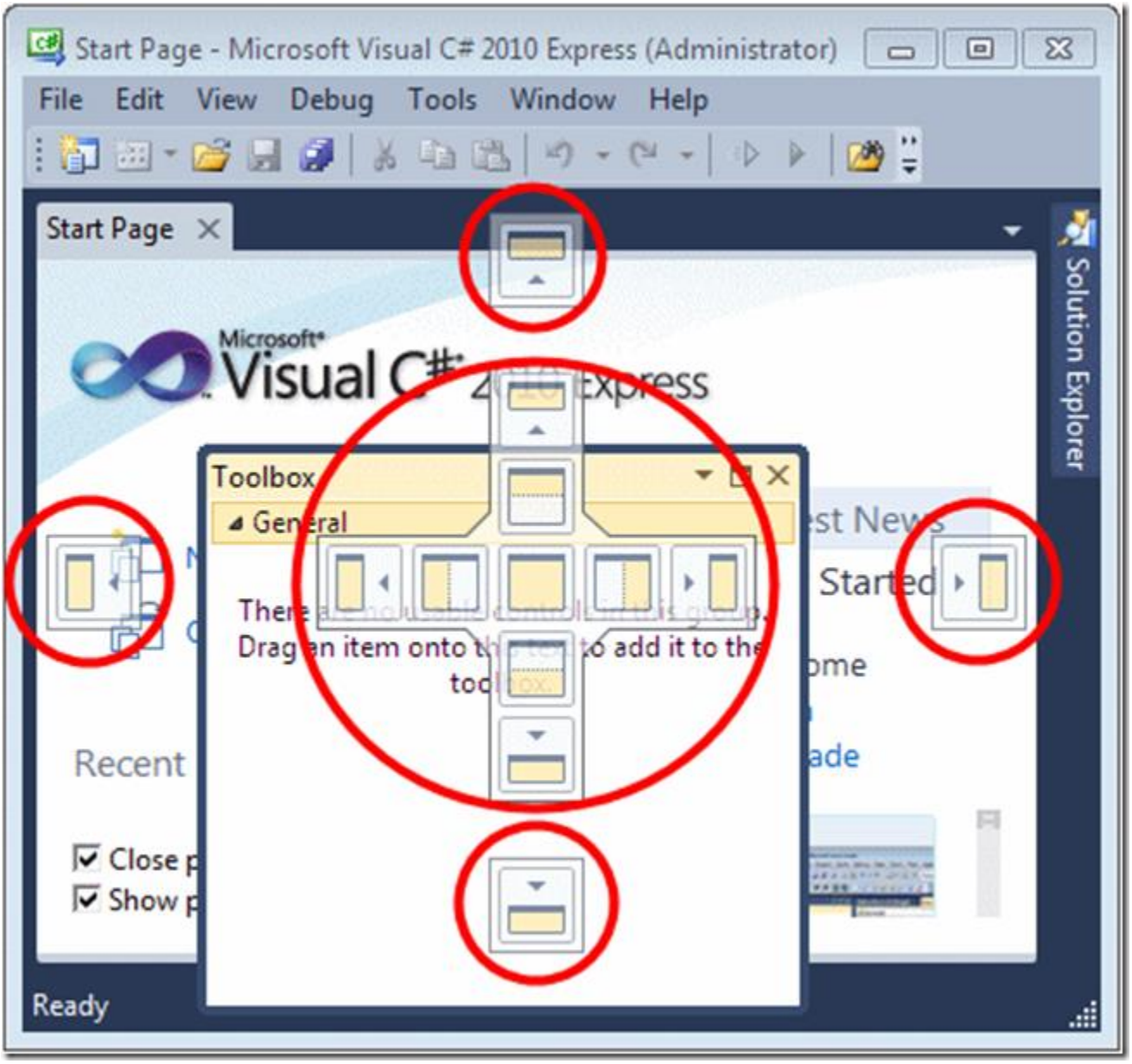
چنانچه پس از باز کردن پنجره متوجه شدید که دیگر به آن نیازی نیست، می توانید با برداشتن مکان نمای موس از روی آن، پنجره را محو کنید. پنجره ی نام برده به حالت پیشین خود باز می گردد. اما، چنانچه مایلید پنجره ای (حتی پس از برداشتن مکان نمای موس از روی آن) باز بماند باید روی دکمه ی Auto hide کلیک کنید.



به صورت پیش فرض، برنامه برخی از پنجره ها را در سمت چپ و برخی دیگر را در سمت راست صفحه قرار می دهد. همچنین، می توان آرایش و ترتیب پنجره ها را مطابق میل تغییر داد. برای این منظور، روی نوارعنوان (title bar) پنجره کلیک کرده و آن را در جهت موقعیت مورد نظر خود بکشید. همان طور که در تصویر مشاهده می کنید، برنامه پنج موقعیت مختلف را به تصویر می کشد.

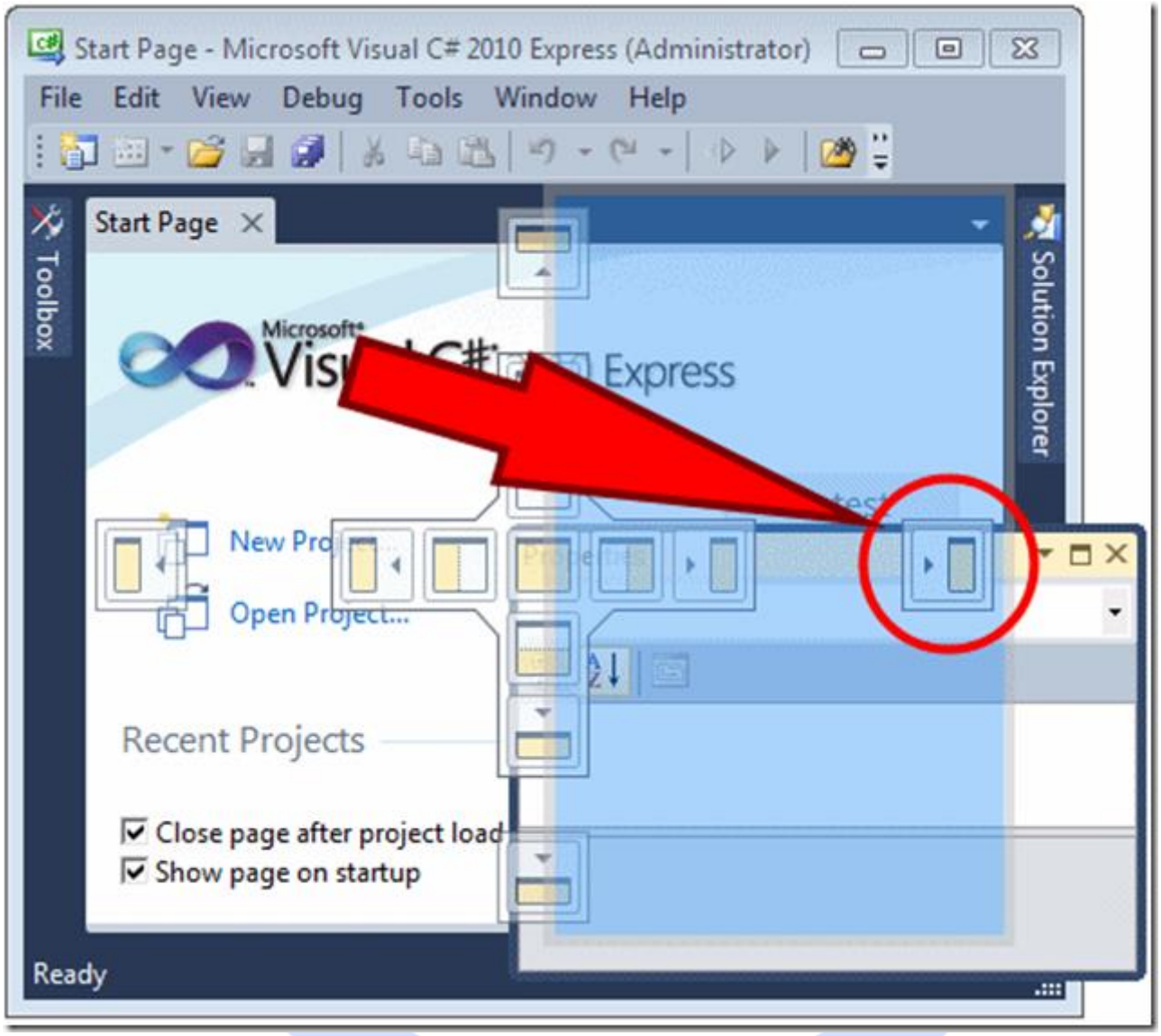
آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

قرار دادن پنجره در کناره های برنامه (محیط ویژوال استودیو)



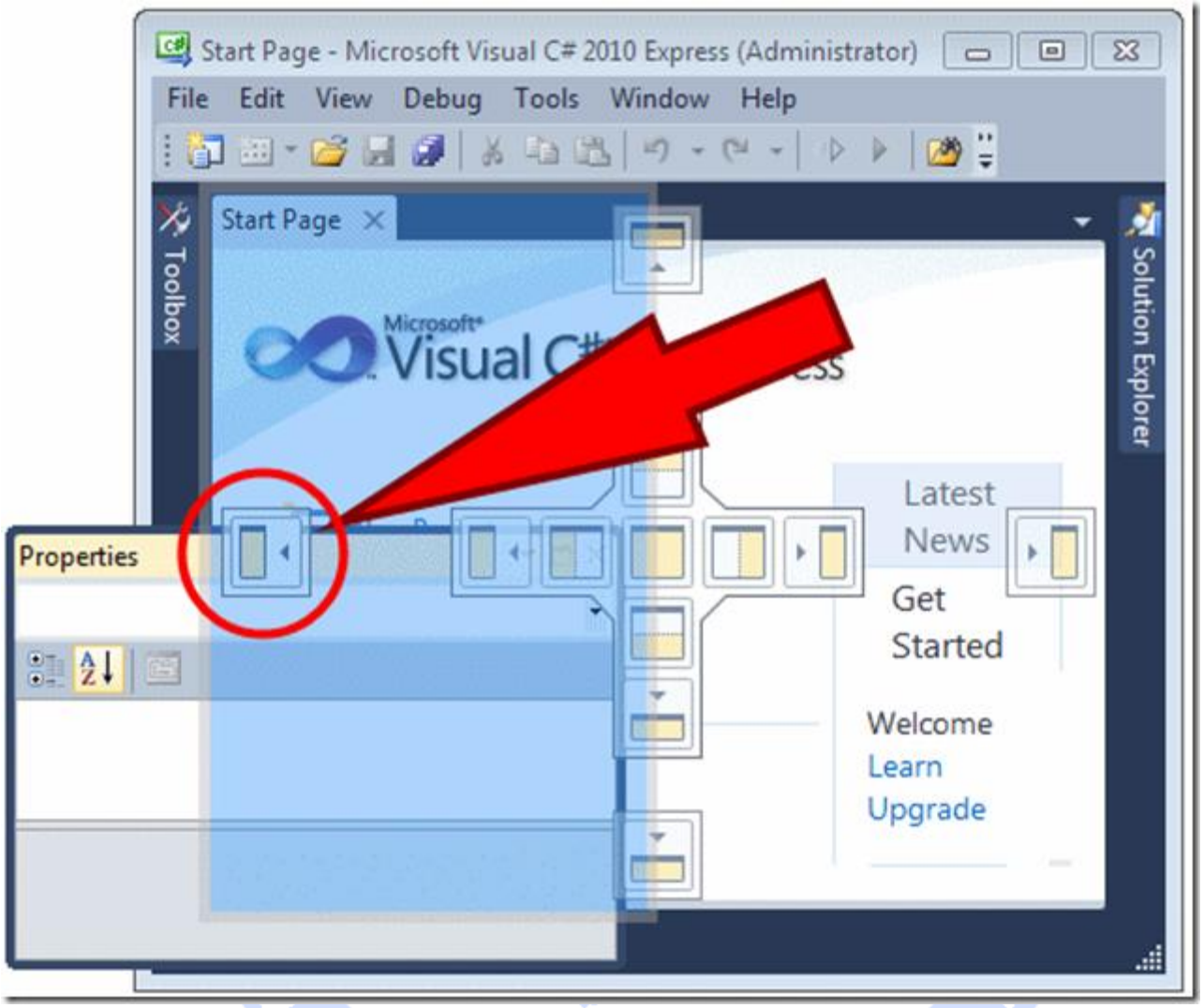
برای قرار دادن پنجره در کناره های برنامه، باید کلیک چپ را روی نوارعنوان پنجره نگه داشته و آن را به ناحیه ی موردنظر بکشید. حال، مستطیلی آبی رنگ ظاهر می شود که ناحیه ی مورد نظر را نشان می دهد. برای قرار دادن پنجره در دورترین ناحیه ی سمت راست یا چپ، پنجره را در آن قسمت رها کنید. به مثال زیر توجه کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



یا

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

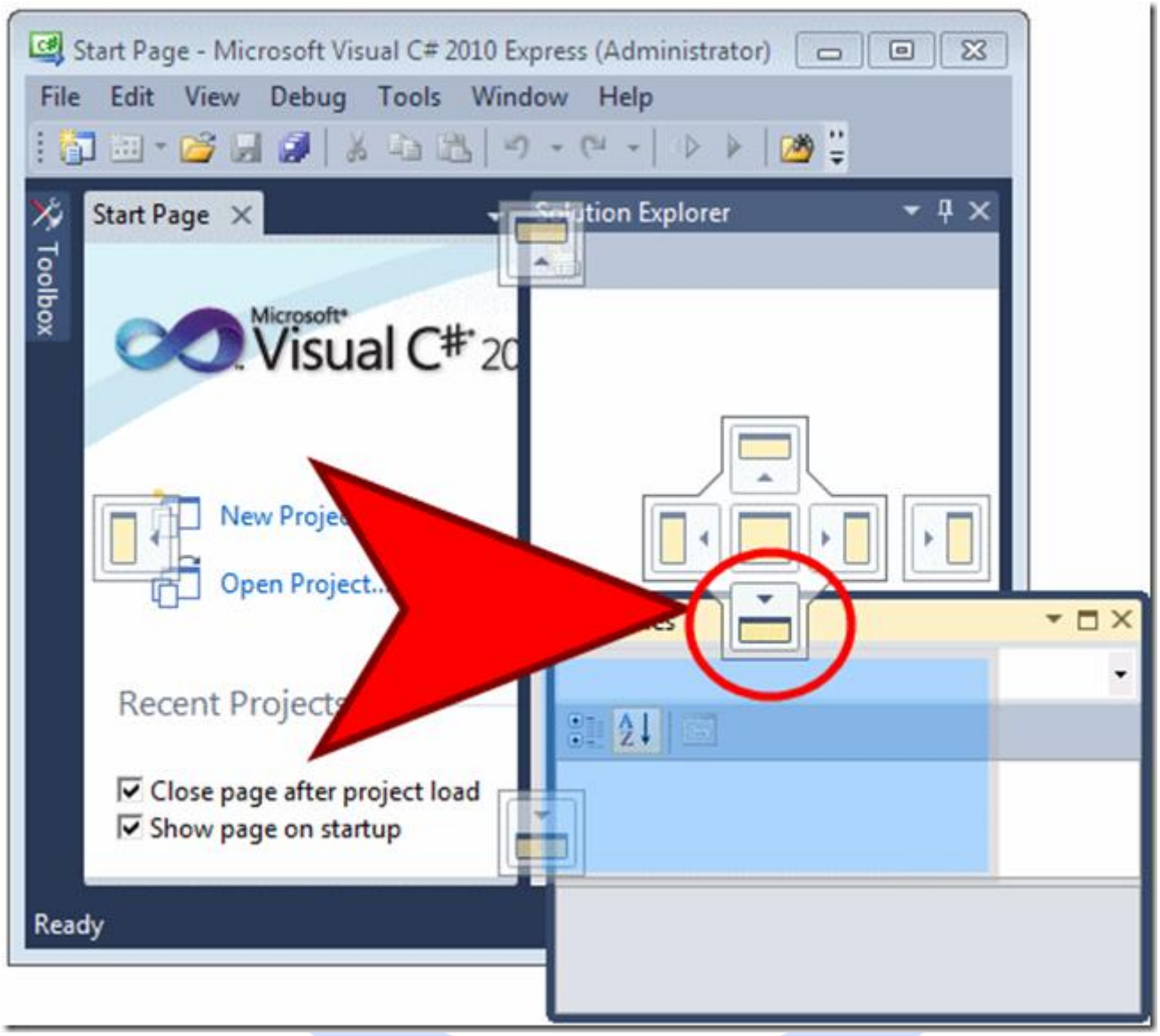


برای قرار دادن پنجره در بالاترین یا پایین ترین قسمت صفحه نیز مانند دستورالعمل ذکر شده عمل کنید.

قرار دادن پنجره ای در بالا یا پایین پنجره ای دیگر

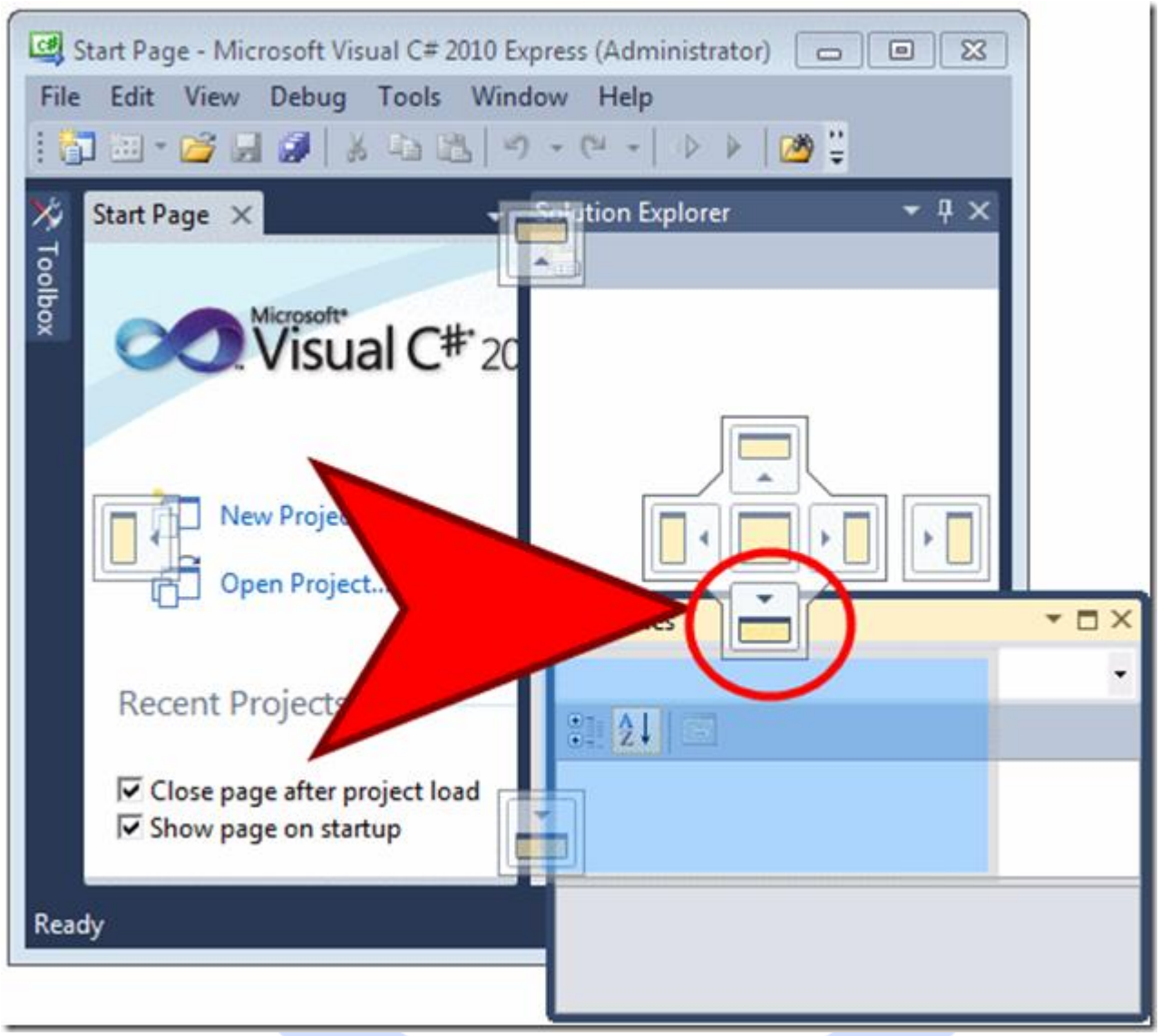
برای داشتن دو پنجره در یک صفحه، باید پنجره ی جدید را در بالا یا پایین پنجره ی موجود قرار دهید. ابتدا، باید پنجره ای به عنوان پنجره ی مرجع یا اصلی داشته باشید،سپس پنجره ی دیگری را به پنجره ی موجود اضافه کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



به این تبدیل می شوید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



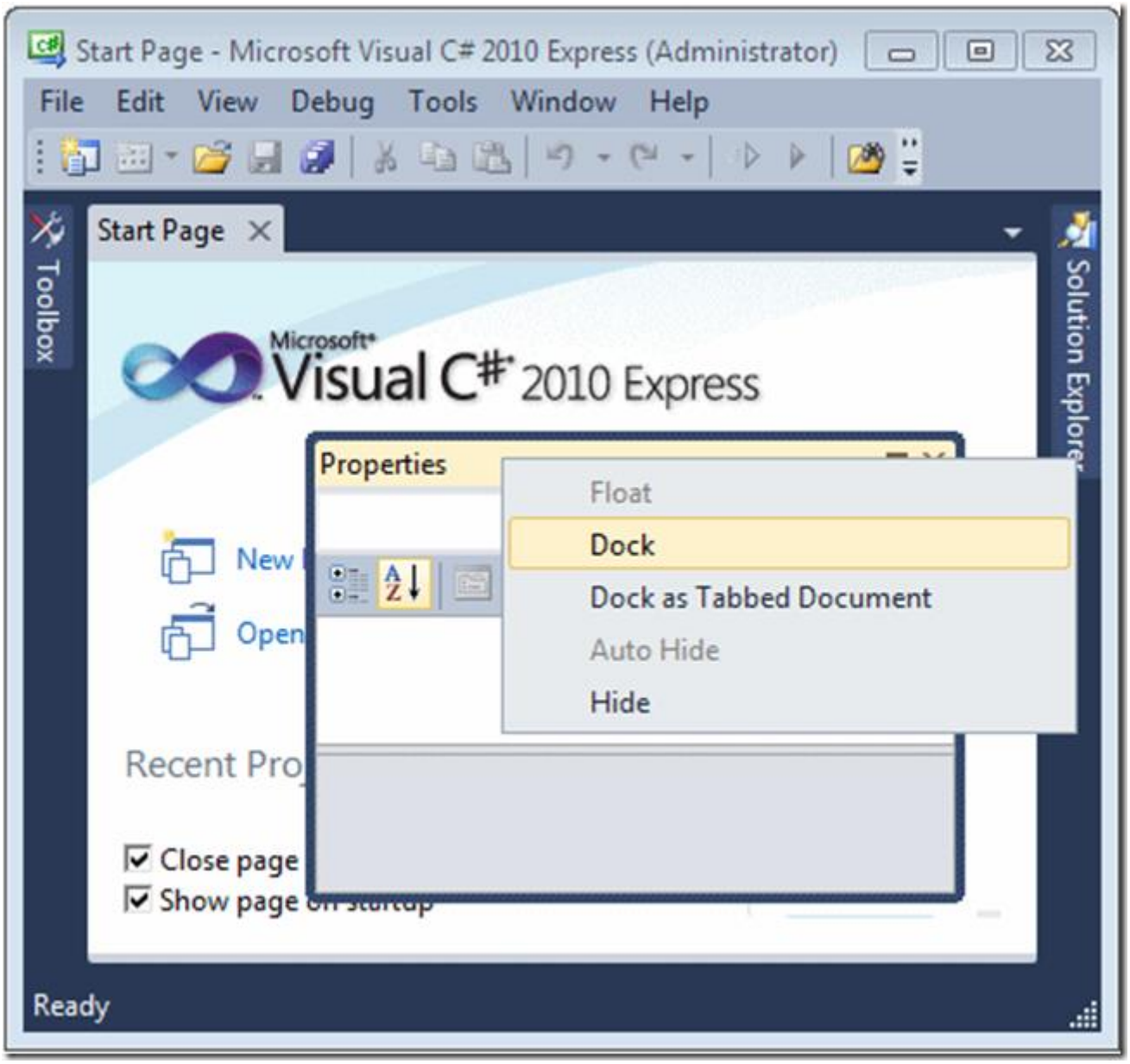
پنجره های شناور

بیشتر پنجره ها در یک طرف صفحه قرار می گیرند. با این وجود، می توان پنجره ای را روی پنجره های دیگر قرار داد. به چنین پنجره ای شناور می گویند. برای این منظور، روی نوارعنوان پنجره کلیک چپ را نگه داشته و بکشید، سپس آن را در جایی وسط صفحه رها کنید.

پنجره ی شناور گزینه ای دارد به نام دکمه ی Maximize (بزرگنمایی). اگر روی این دکمه کلیک کنید، پنجره تمام صفحه نمایش را اشغال می کند. این کار را می توانید با دوبار کلیک روی نوارعنوان نیز انجام دهید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

چنانچه پنجره ای شناور است و شما مایلید آن را در جای مشخصی از صفحه (مثلاً سمت راست) ضمیمه / متصل کنید، باید روی نوارعنوان (title bar) کلیک راست کرده و گزینه ی Dock را انتخاب کنید.

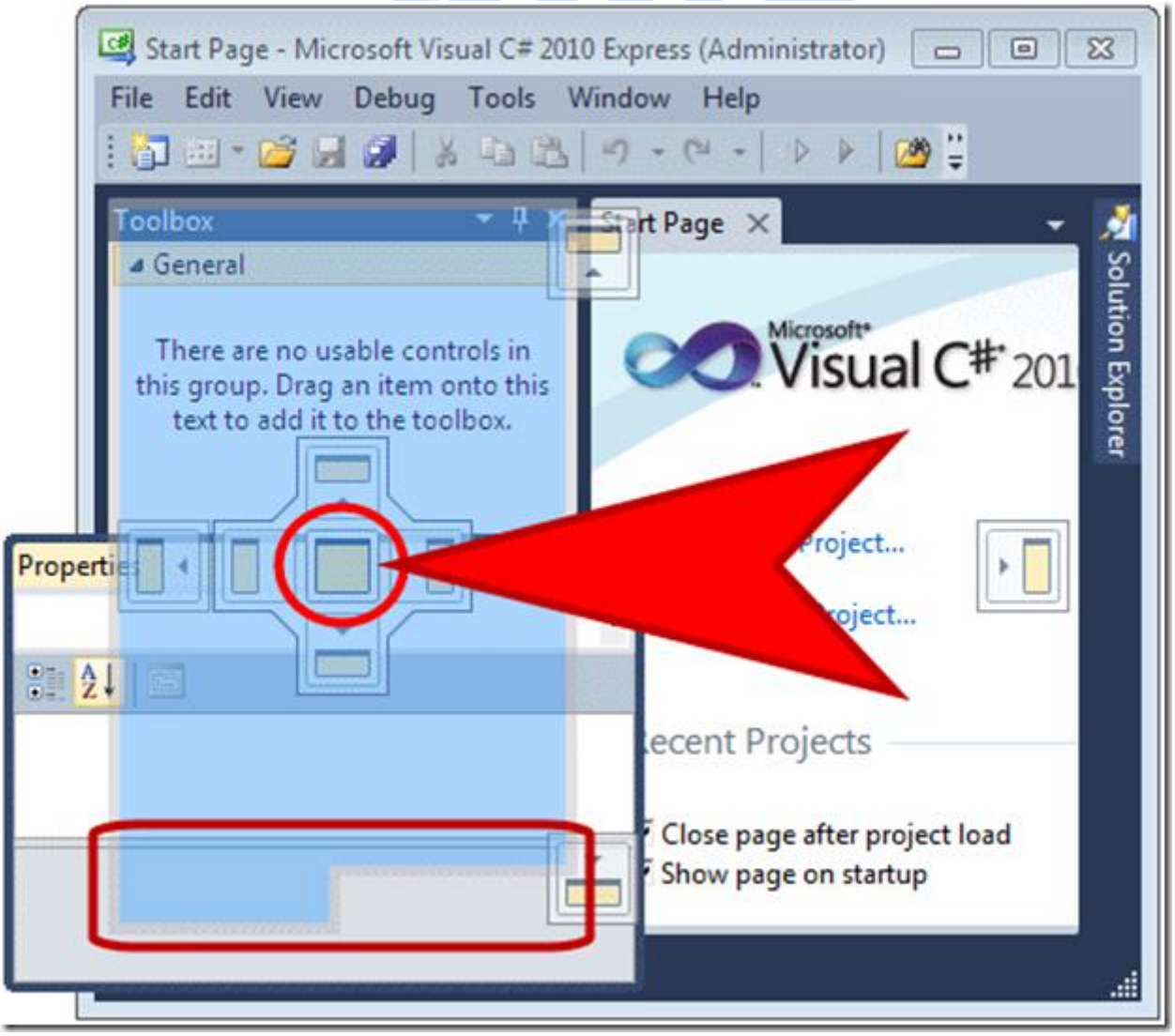


چناچه، پنجره ای به ناحیه ی خاصی از صفحه متصل است و می خواهید آن را شناور کنید، روی نوارعنوان کلیک راست کرده و گزینه ی Float را انتخاب کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

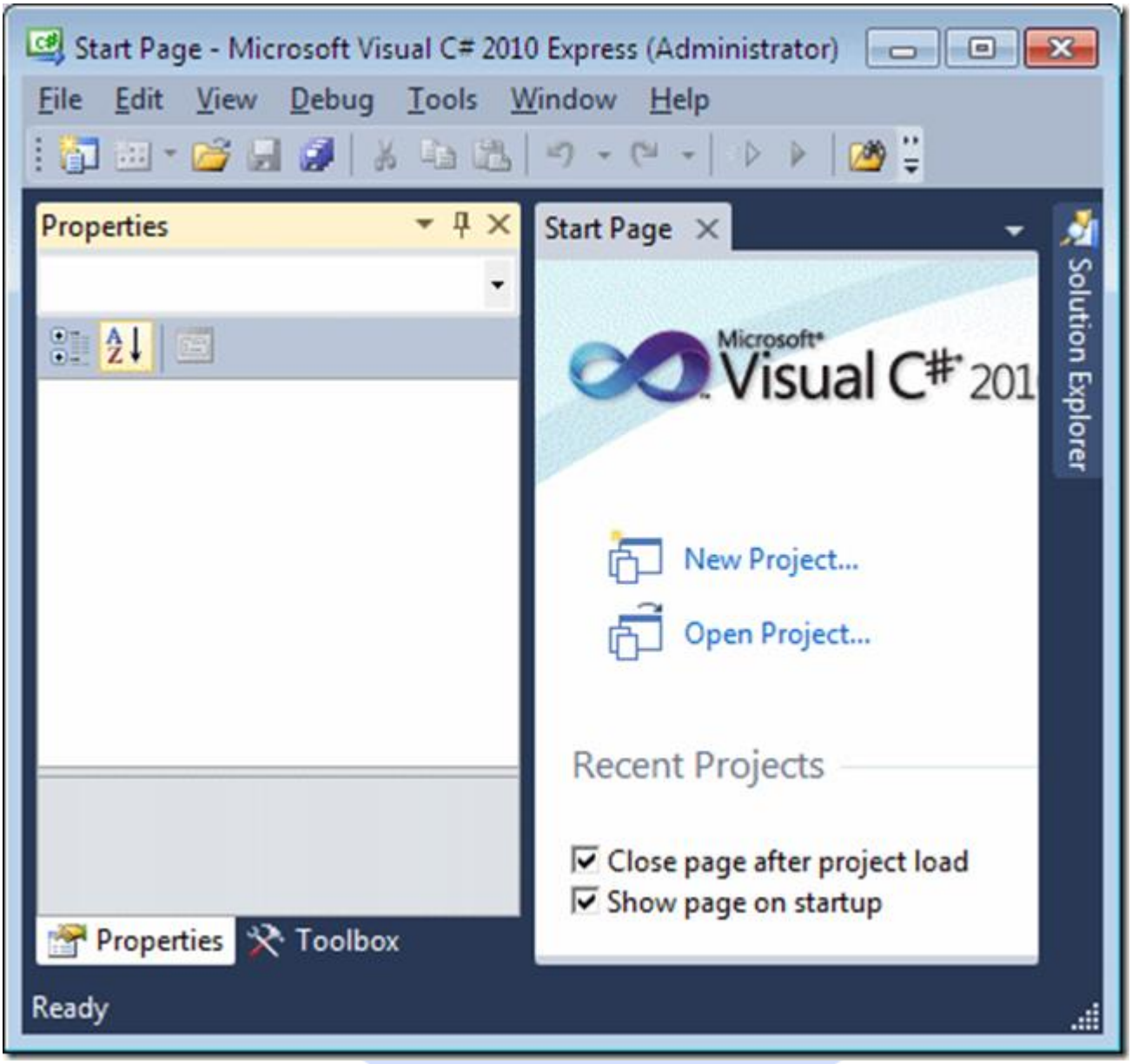
قرار دادن چند پنجره در یک ناحیه

می توان در یک طرف صفحه، چند پنجره را باهم گنجاند. ابتدا، پنجره ی پایه (اصلی) را بزرگ نمایی کرده و آن را به ناحیه ی مورد نظر متصل کنید. سپس، نوارعنوان پنجره را گرفته و آن را به سمت ناحیه ی مقصد بکشید، تا این که قسمت پایین پنجره ی اصلی دو بخش برجسته/پررنگ و غیر برجسته پیدا کند.



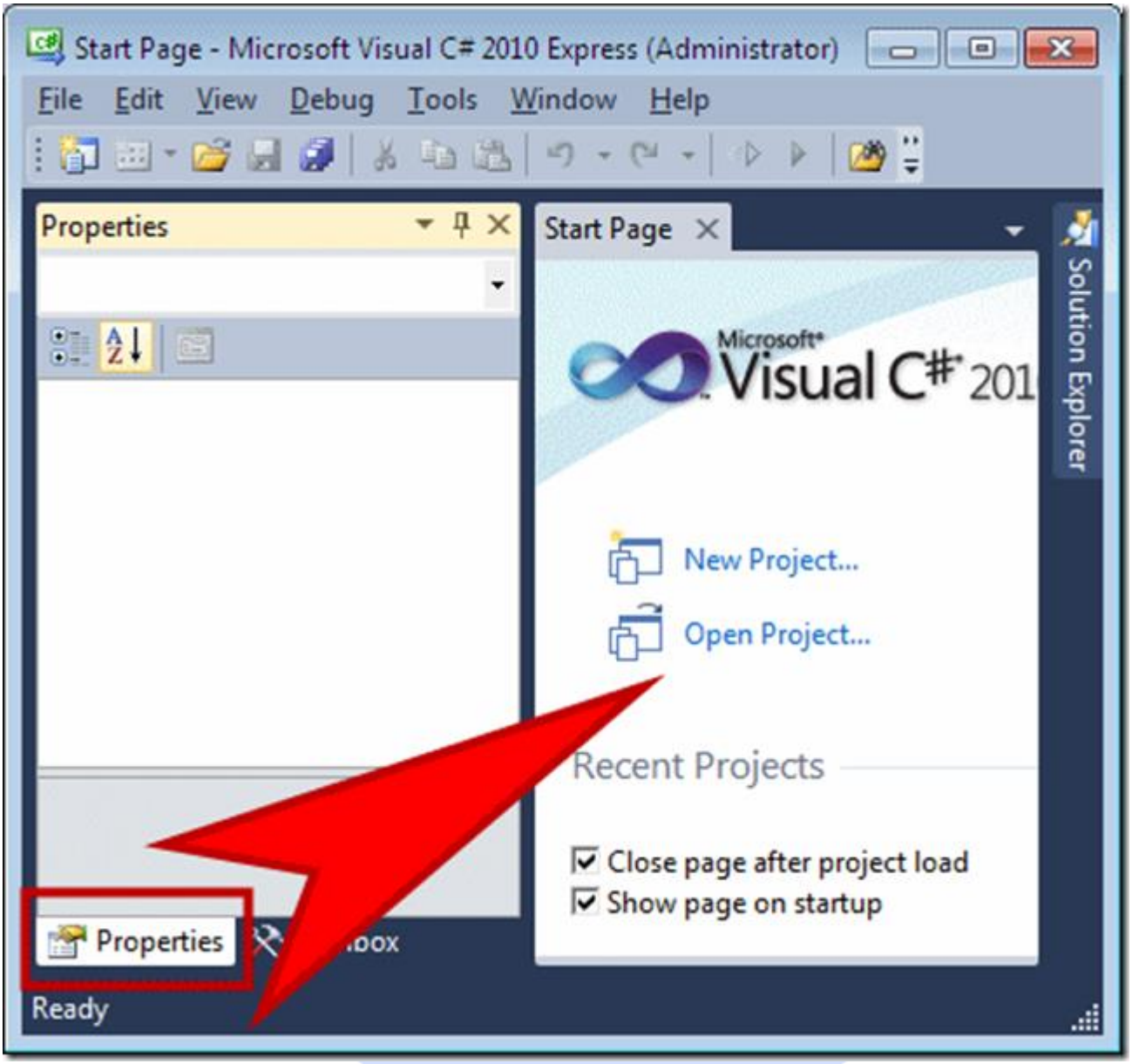
حال موس را رها کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



به منظور حذف پنجره، تب آن را بیرون بکشید.

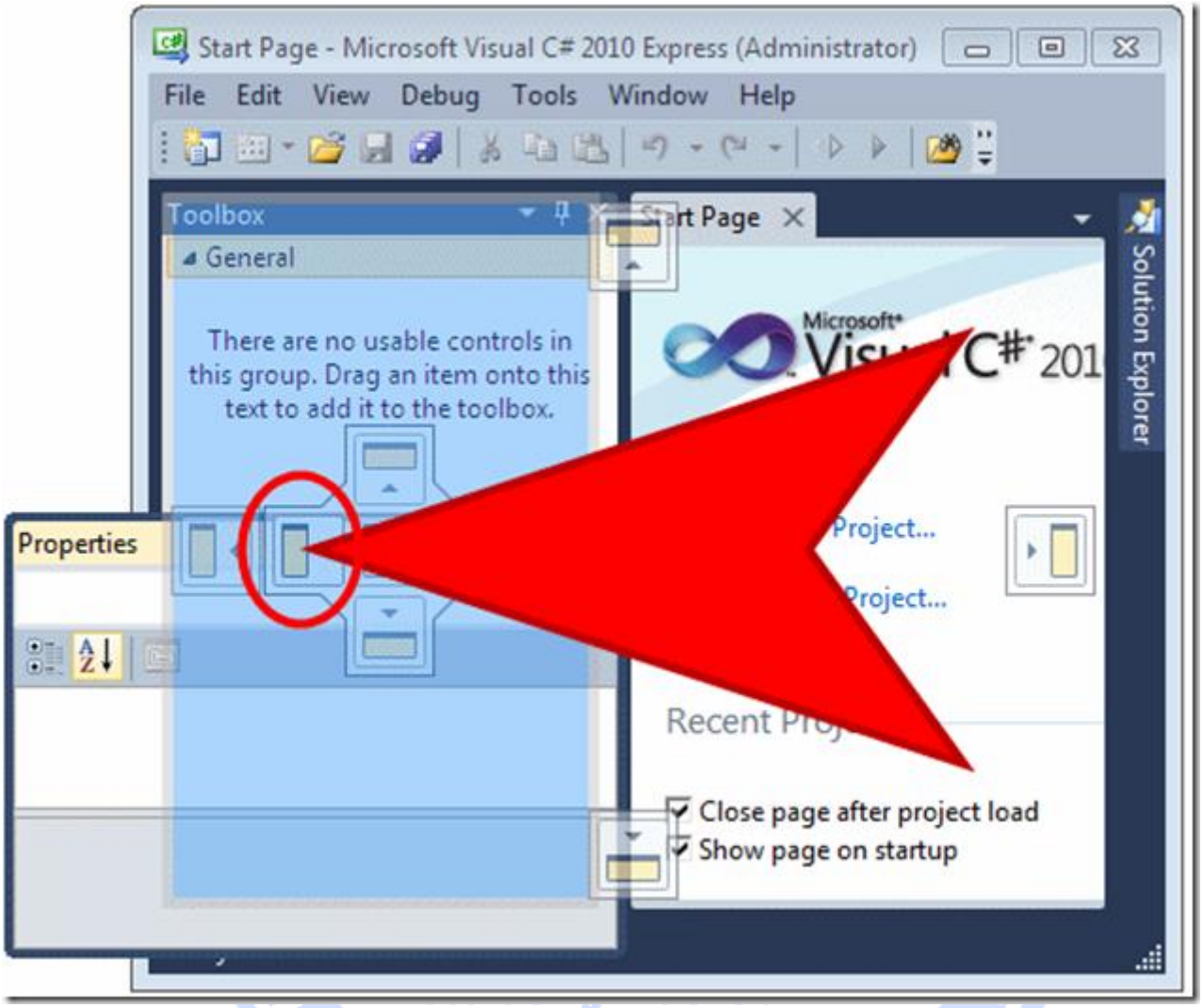
آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



برای قرار دادن پنجره درسمت راست یا چپ پنجره ی موجود، ابتدا پنجره ی فرعی را به محل پنجره ی اصلی (موجود) بکشید. سپس، کادر راهنما (guiding box) ی سمت چپ را انتخاب کنید.

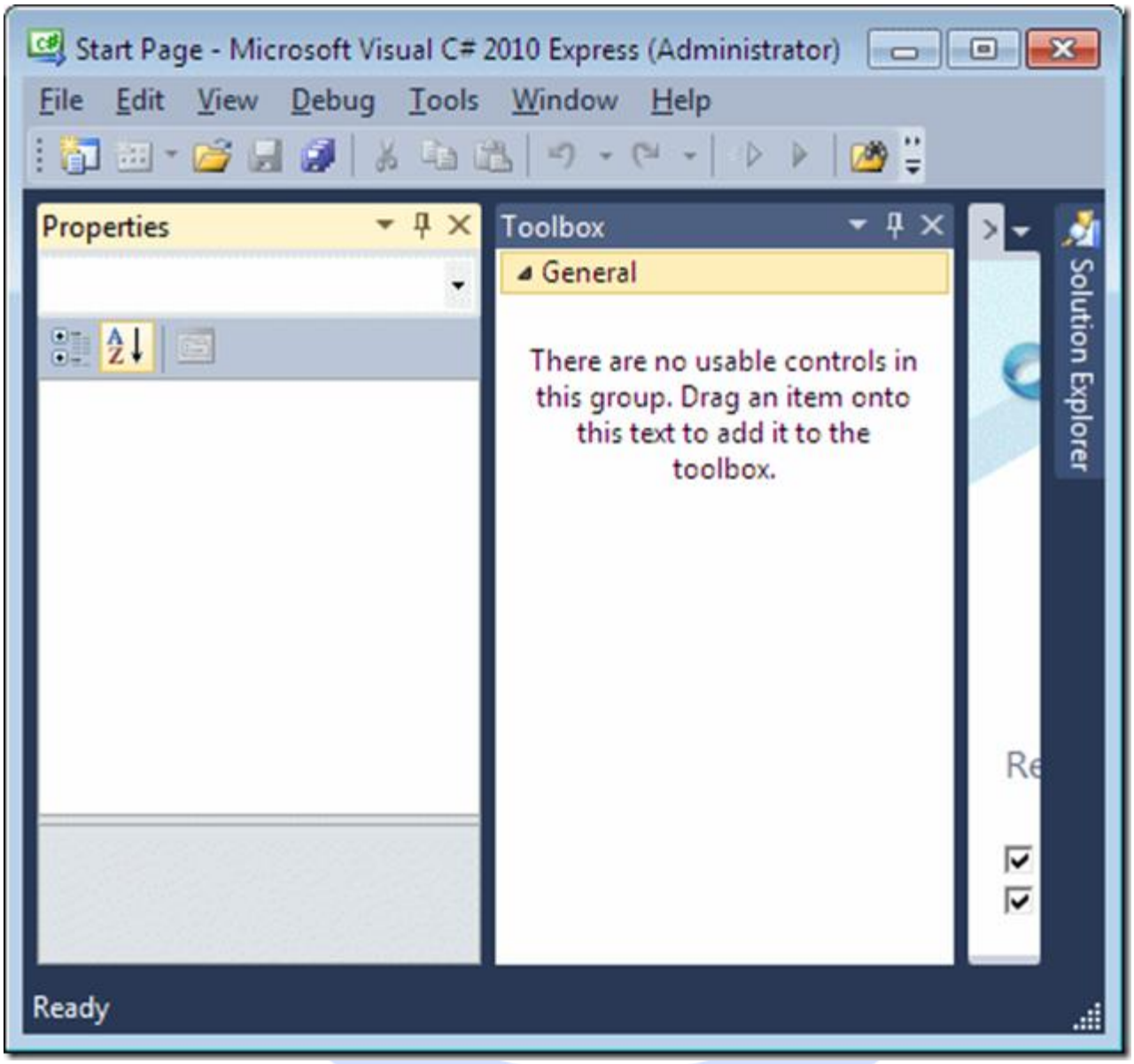
این

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



به صورت زیر در می آید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

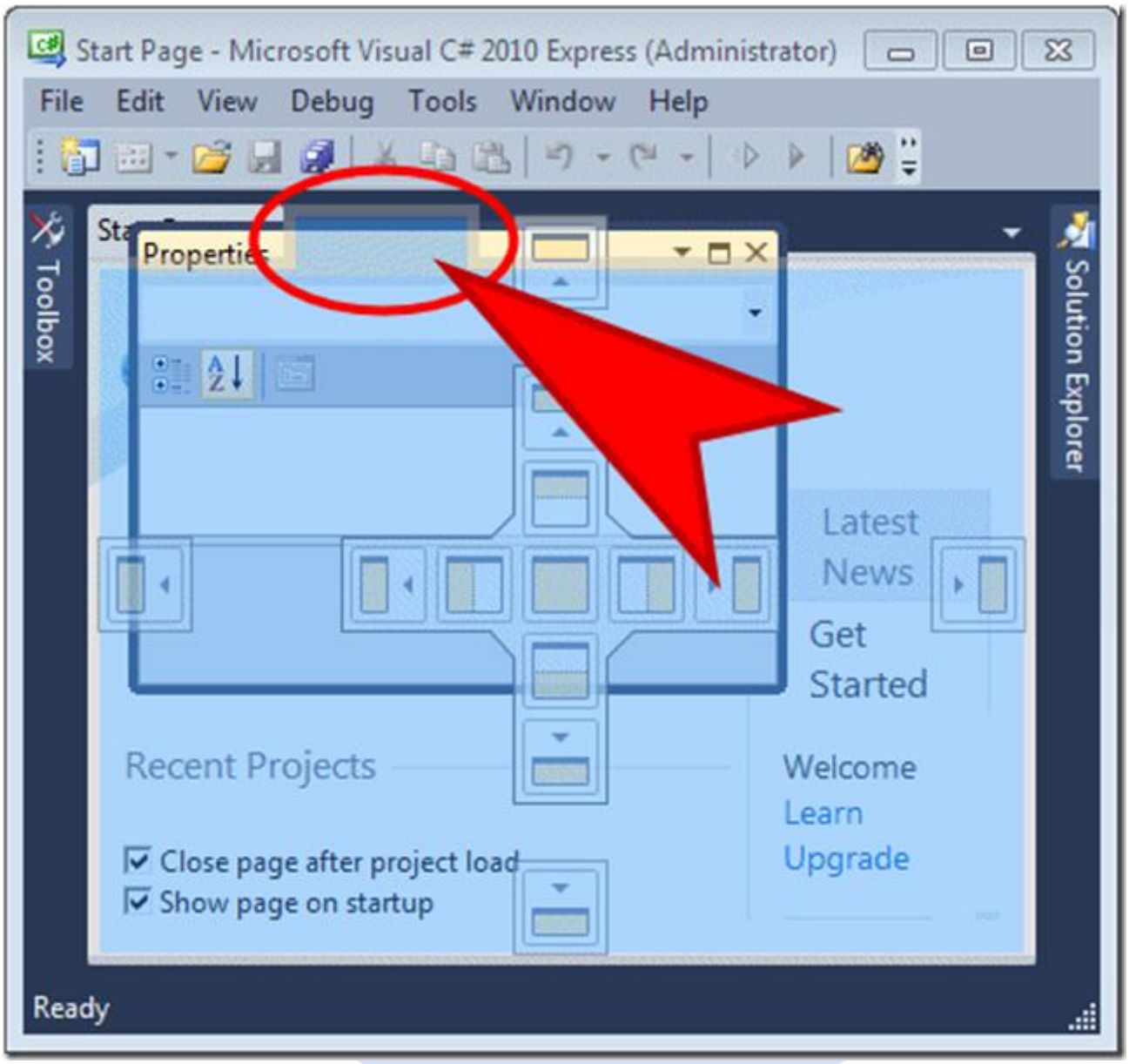


پنجره های تب دار (Tabbed Windows)

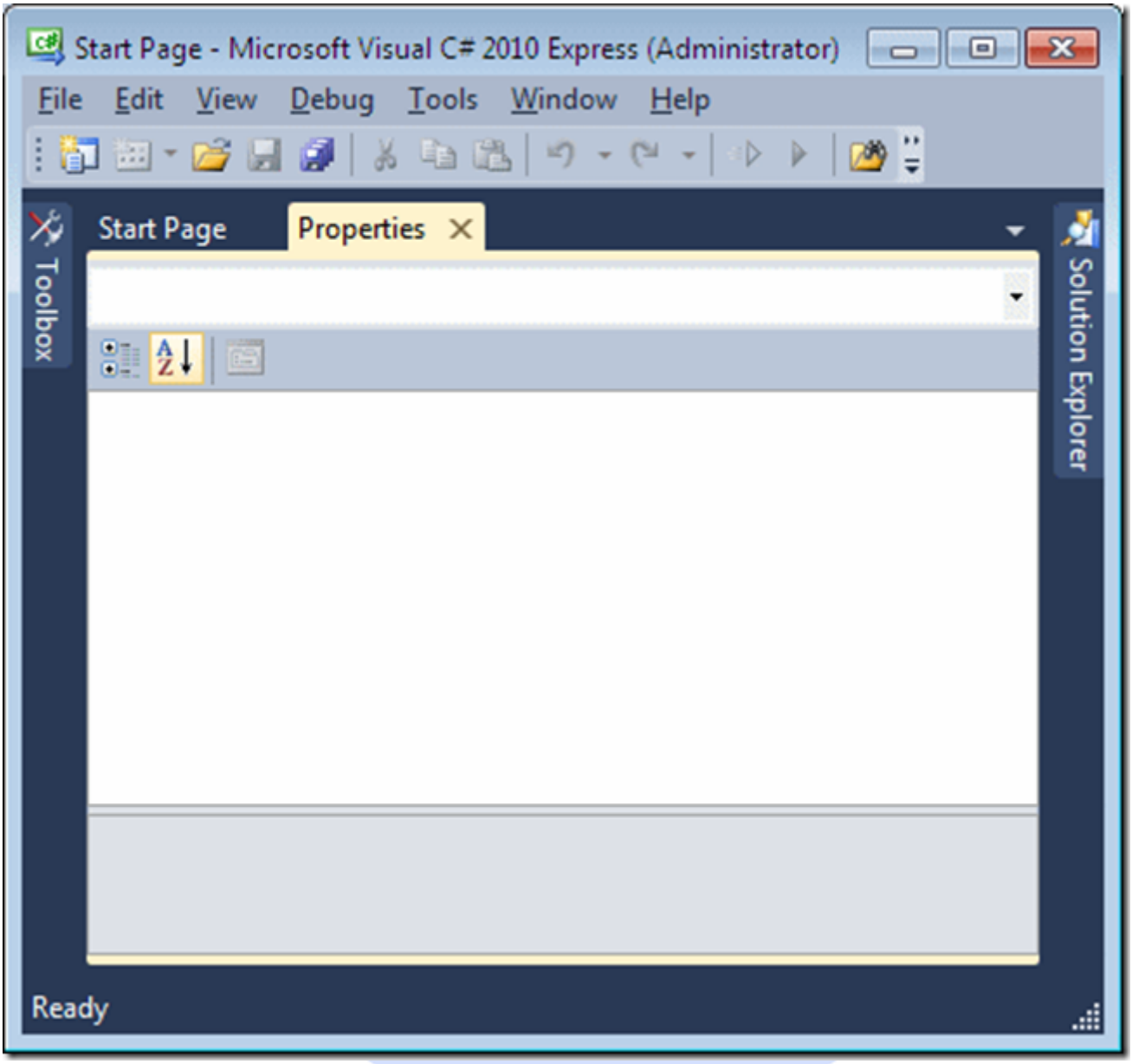
می توانید کاری کنید که پنجره تب خود را در بالاترین قسمت Code editor نشان دهد. این کار را می توان به دو صورت انجام داد.

می توان پنجره را کشید و آن را روی تب Start page رها کرد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



می توان ابتدا راست کلیک کرد و بعد گزینه ی Dock as Tabbed Document را انتخاب کرد.

از زبان برنامه نویسی برای دادن دستور انجام کار خاصی به کامپیوتر استفاده می شود. زبان برنامه نویسی مانند زبان محاوره از کلمات معنی دار، گرامر، فرمول ها و قوانین خاص تشکیل شده که برای ساختن جملات و مفاهیم معنادار باید از آن قوانین پیروی کرد.

زبان های برنامه نویسی متعددی وجود دارند. برخلاف زبان انسان ها، زبان های برنامه نویسی کامپیوتر بسته به اهداف یا نقش هایی که ایفا می کنند، طبقه بندی می شوند.

سخت افزارها عبارتند از مجموعه قطعاتی که دستور را از زبان برنامه نویسی دریافت می کنند. برخی زبان های برنامه نویسی برای بخش (سخت افزاری) خاصی تعبیه شده اند و یا محیط خاصی را هدف دستورات خود قرار می دهند. چنین محیطی ، از سیستم عامل خاصی استفاده می کند که از یک وسیله (مثل موبایل) به وسیله ی دیگر (کامپیوتر) قابل انتقال نمی باشد. برخی زبان های برنامه نویسی هم بخش یا قطعه ی خاصی را هدف نمی گیرند.

برخی زبان ها برای رفع یک مسئله ی خاص طراحی شده اند و توانایی انجام وظایف معمول (زبان های دیگر) را ندارند. برای مثال، زبان هایی وجود دارند که فقط قادر به نشان دادن محتویات وب هستند و از آن ها نمی توان برای انجام محاسبات ریاضی و هندسی بهره گرفت. برخی برنامه ها طوری تعبیه شده اند که تنها با مقدارهای (value) خاصی مثل (true یا false) کار می کنند. تعدادی زبان برنامه نویسی هم داریم که به حل تمام مسائل می پردازند.

C# از آن دسته زبان هایی است که علاوه بر اجرای دستورات، به حل مسائل نیز می پردازد. این زبان بر روی تمام سیستم عامل ها از جمله ویندوز قابل اجرا است. C# از جمله زبان هایی است که در Microsoft.NET Framework به کار می رود. Microsoft.NET Framework کتابخانه ای از اشیاء است که به کمک آن می توان در کامپیوتر چیزهایی را به وجود آورد یا ترسیم کرد.

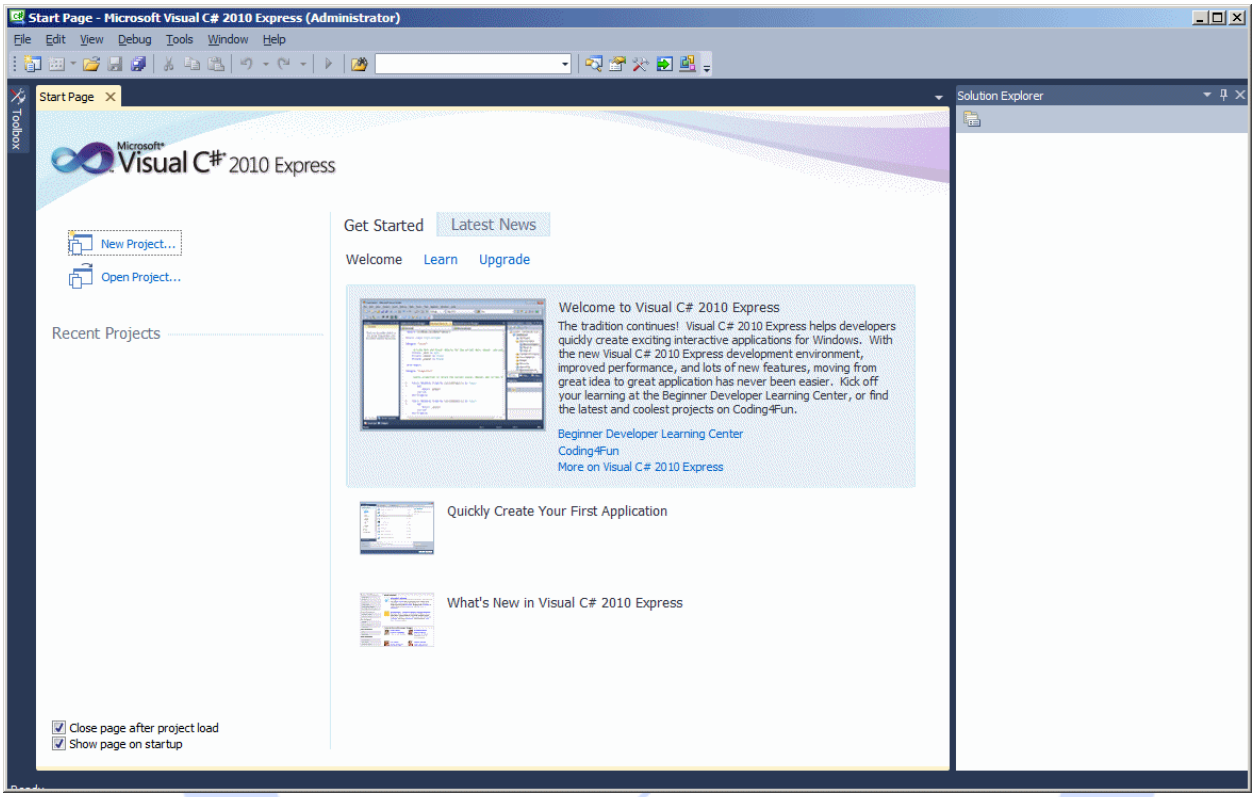
پروژه ی کامپیوتری عبارتند از گروهی فایل و شئ که دستورهای مورد نظر را برای انجام عملکرد های معین تعبیه می کند. پروژه ی C# متشکل از چند فایل است که دستورهای برنامه را دربردارند. یک پروژه ی ساده ی C# دربردارنده ی یک فایل است که تمام دستورات مورد نیاز برنامه را حمل می کند. این در حالی است که پروژه های ی پیشرفته C# حاوی بیش از یک فایل است.

برنامه های کاربردی کنسول

زبان برنامه نویسی C# به منظور طراحی و نوشتن برنامه های کاربردی DOS به کار می رود. برای طراحی چنین برنامه ای باید دستورات مورد نظر را در برنامه ی Notepad یا محیط برنامه نویسی مانند visual studio نوشت.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

برای راه اندازی برنامه Microsoft Visual C# 2010 Express، دستورالعمل زیر را دنبال کنید : Start -> (All) Programs -> Microsoft Visual Studio 2010 Express -> Microsoft Visual C# 2010 Express



راه اندازی پروژه

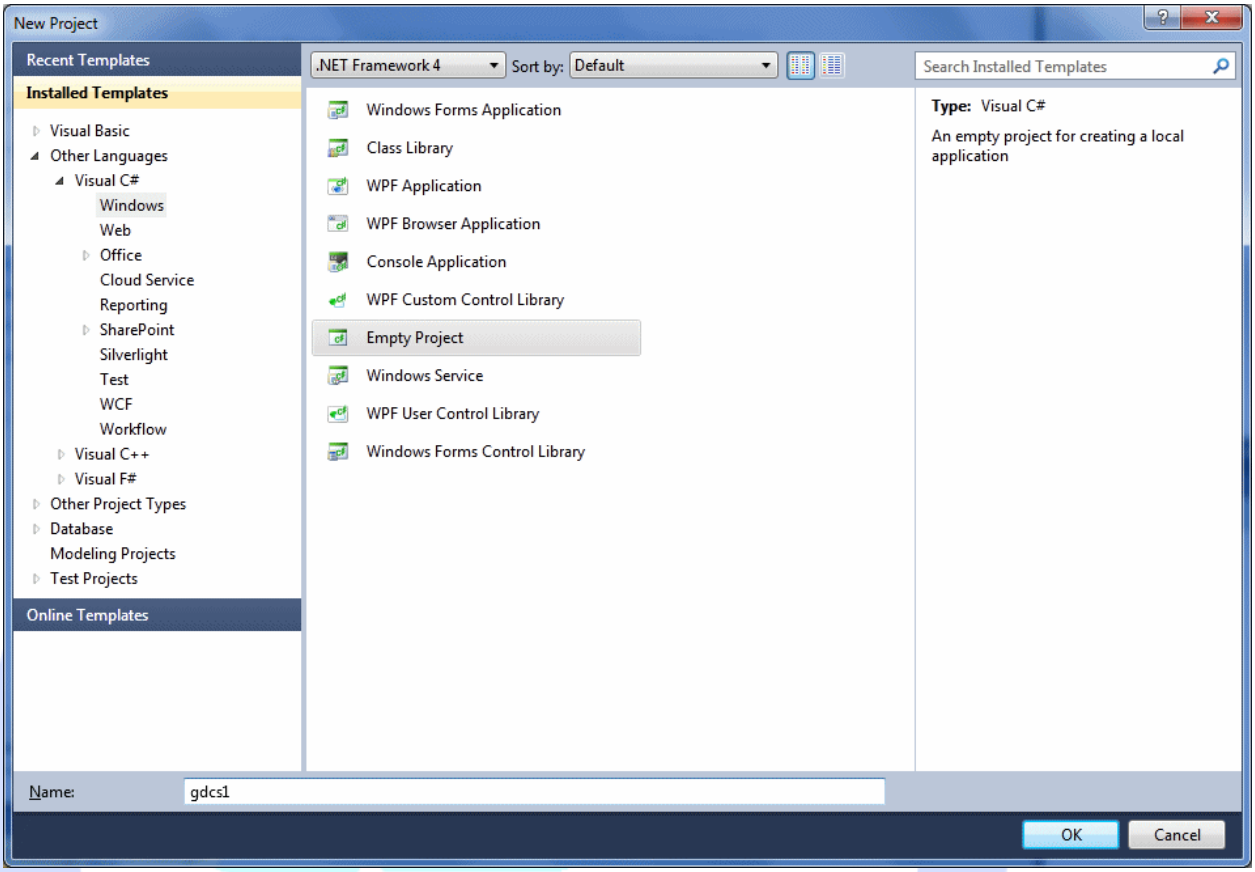
برنامه ی Microsoft Visual C# 2010 Express را اجرا کنید.

به منظور ایجاد برنامه ی کاربردی جدید (app)، در فهرست گزینه ی اصلی (main menu) روی گزینه ی File -> New Project... کلیک کنید.

در لیست وسطی نیز گزینه ی Console Application را کلیک کنید.

اسم مورد نظر را به gdcs1 تغییر دهید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



روی گزینه ی ok کلیک کنید.

ایجاد فایل برای کد مورد نظر

هدف از نوشتن برنامه، دادن دستور (اجرای عملکرد خاص) به کامپیوتر است. البته، این برنامه را با زبان عادی (مثلاً انگلیسی) و جملات معنی دار می نویسیم. به عبارت دیگر، یک دستور معمولی از متن ساده استفاده می کند (که شامل حروف الفبا، ارقام و علامت های ناخوانا می باشد).

می توان این دستورات را با کمک برنامه های ویرایشگر متن (text editor) از قبیل Notepad، Microsoft word، wordpad و wordperfect نیز نوشت. در برنامه نویسی، قوانین و دستور هایی وجود دارد که پیروی از آن ها الزامی است. به دستورات مزبور کد هم گفته می شود.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

در برنامه های Microsoft Visual C# Express و Microsoft Visual Studio Professional، ویرایشگر متنی وجود دارد که از آن با نام code editor یاد می شود. چناچه، پروژه ی خود را با فرمت console application بنویسید، فایل پیش فرضی به وجود می آید که می توان آن را تنظیم (customize) کرد. اما اگر برنامه ی خود را به عنوان پروژه ای خالی آغاز کنید، لازم است خود یک فایل به آن اضافه کنید. برای این منظور دستورات زیر را انجام دهید.

Main menu را باز کرده و روی گزینه های Project -> Add New Item… کلیک کنید.

حال، در قسمت solution explorer، روی اسم پروژه ی مورد نظر راست کلیک کرده و مکان نمای موس را روی گزینه ی Add قرار دهید، سپس New Item را انتخاب کنید.

هریک از دستورهای مذکور، پنجره ی محاوره Add new item را نشان می دهد. حال، در لیست میانی، روی گزینه ی Code File کلیک کنید. می توانید اسم فایل مذکور را بپذیرید یا آن را مطابق میل تغییر دهید. پس از کلیک روی گزینه ی ok، پنجره ی سیاهی نمایان می گردد. به همین شیوه، می توانید هر تعداد فایل که مایلید به پروژه اضافه کنید.

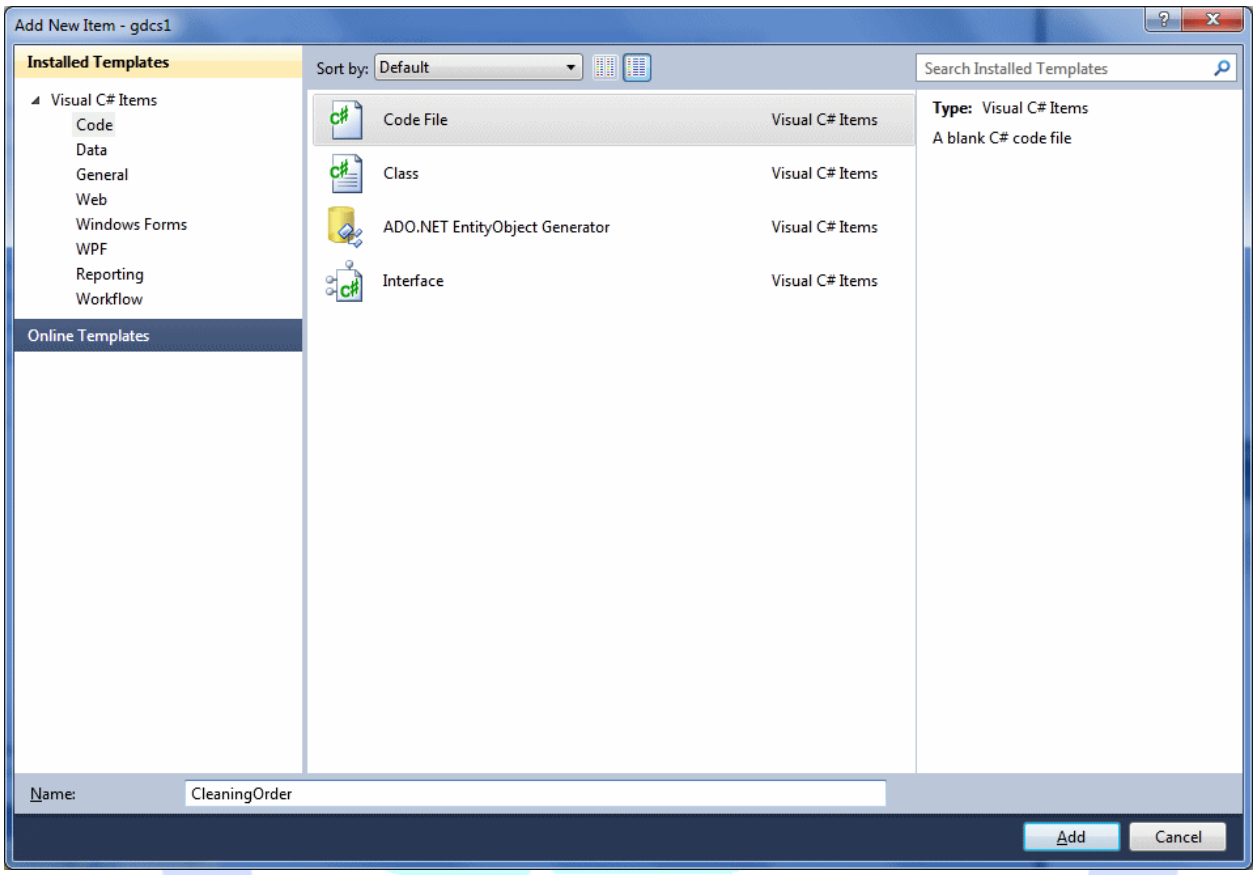
ایجاد فایل

به منظور افزودن فایل جدید به کد مورد نظر، به فهرست گزینه ی اصلی (main menu) مراجعه کرده و روی Project -> Add New Item… کلیک کنید.

در صورت استفاده از visual studio، باید آیتم های visual C# (در لیست سمت چپ) را بزرگ نمایی کرده و روی code کلیک کنید.

اسم موردنظر را به CleaningOrder تغییر دهید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



اکنون، گزینه ی Add را انتخاب کنید.

کد اصلی C#

به منظور ایجاد ساده ترین نوع برنامه ی کاربردی C#، کافی است تنها یک کد پایه (basic) بنویسید. کد مذکور به این صورت است.

```
class Exercise
{
  static void Main(string[] args)
  {
    System.Console.WriteLine("Hello world");
  }
}
```

کد نام برده بخش های مختلفی دارد که در مباحث بعدی به هریک از آن ها مفصل پرداخته می شود.
کلید واژه ی class در درس ۵ معرفی می شود.

واژه ی Exercise تنها یک اسم است. درباره ی اسم ها و تمام قوانین مربوط به آن در درس بعدی بحث می کنیم.

علامت های ()، {} در درس ٤ معرفی می شوند.

نحوه ی استفاده از کلیدواژه ی static در درس ۱۰ بحث می شود.

معنی کلیدواژه ی void در درس ٥ توضیح داده می شود.

Main را در درس ۷ معرفی می کنیم و در درس ۲٦ مفصل توضیح می دهیم.

نقش واژه ی system در درس ۸ تشریح می شود.

نقش کاراکتر نقطه "." را درس ٥ تشریح می کنیم.

Console و waterline در درس ٦ توضیح داده می شوند.

علامت (") را در درس ٤ معرفی می کنیم و مفصل در درس ۲۸ تشریح می کنیم.

این کد را تا حد امکان ساده معرفی کرده ایم. تک تک کلمات و عملگرها در این کد وظیفه ی خاصی را انجام می دهد ولی می توان آن ها را به شیوه های مختلف به کاربرد.

نوشتن کد اصلی

کد زیر را در داکیومنت خالی بنویسید.

```
class Order
{
  static void Main(string[] args)
  {
    System.Console.WriteLine("Georgetown Dry Cleaning Services");
    System.Console.ReadKey();
  }
}
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

توضیحات

خط یا پاراگرافی از متن که جزء کد محسوب نمی گردد. در زبان C# دو نوع comment وجود دارد. برای نشان دادن / قرار دادن comment در (خطی از) متن، خط مورد نظر را با علامت // آغاز کنید. هر چه در سمت راست این علامت قرار بگیرد، نادیده گرفته می شود. به مثال زیر توجه کنید.

```
// This line will be ignored. I can write in it anything I want
```

Comment مذکور در تنها یک خط قابل استفاده است. می توان comment را با علامت /* نیز آغاز کرد. این نوع comment به علامت */ ختم می شود. هر چه بین این دو علامت /* و */ قرار گیرد، خوانده نمی شود. با این روش می توان، comment را در چند خط (بیش از یک خط) به کار برد. به منظور قرار دادن (افزودن) comment در خط، ابتدا روی خط مورد نظر کلیک کنید. حال، در (قسمت) Standard toolbar، گزینه ی Comment Out the Selected Lines را انتخاب کنید. برای افزودن comment به خط های مجاور، روی متن خط های نام برده راست کلیک کرده، سپس روی گزینه ی Comment Out the Selected Lines کلیک کنید. به منظور حذف comment ها، ابتدا روی متن آن کلیک کنید. سپس، گزینه ی Uncomment the Selected Lines را از Standard toolbar انتخاب کنید.

ایجاد توضیحات

به منظور ایجاد comment، فایل مورد نظر را به صورت زیر تغییر دهید.

```
// Application Name: Georgetown Dry Cleaning Services
  class Order
  {
    // This application is used to perform processing orders for this business
    static void Main(string[] args)
    {
      /* Always greet the customers before processing an order.
      Check the types of clothes a customer has brought.
      Find out the type of cleaning (starch, etc) the customer wants.
      If there are special needs the customer has, make a note. */
      System.Console.WriteLine("Georgetown Dry Cleaning Services");
      System.Console.ReadKey();
    }
  }
```

مدیریت فایل ها

در برنامه های Microsoft Visual C# 2010 Express و Microsoft Visual Studio، پروژه هایی که متشکل از چند فایل هستند، با ورچسب های معینی (واقع در قسمت بالای code editor) نمایش داده می شوند.

فایل های نام برده در main menu ویندوز نیز قابل مشاهده است.

پنجره Solution Explorer

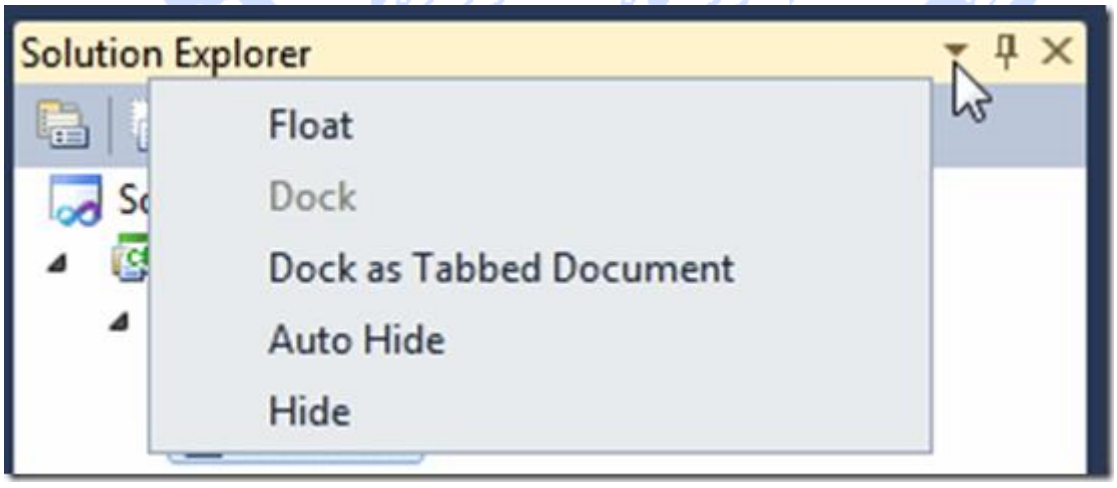
Solution explorer پنجره ای است که فهرستی از فایل های تشکیل دهنده ی پروژه را به نمایش می گذارد. برای دسترسی به پنجره ی مزبور چنانچه، Solution explorer در صفحه ی مانیتور نمایان نیست.

به main menu مراجعه کرده، سپس روی گزینه ی View -> Solution Explorer کلیک کنید.

Standard toolbar را باز کرده، و گزینه ی Solution Explorer را انتخاب کنید.

چنانچه، Solution Explorer در صفحه قابل رویت است، روی تب آن کلیک کنید.

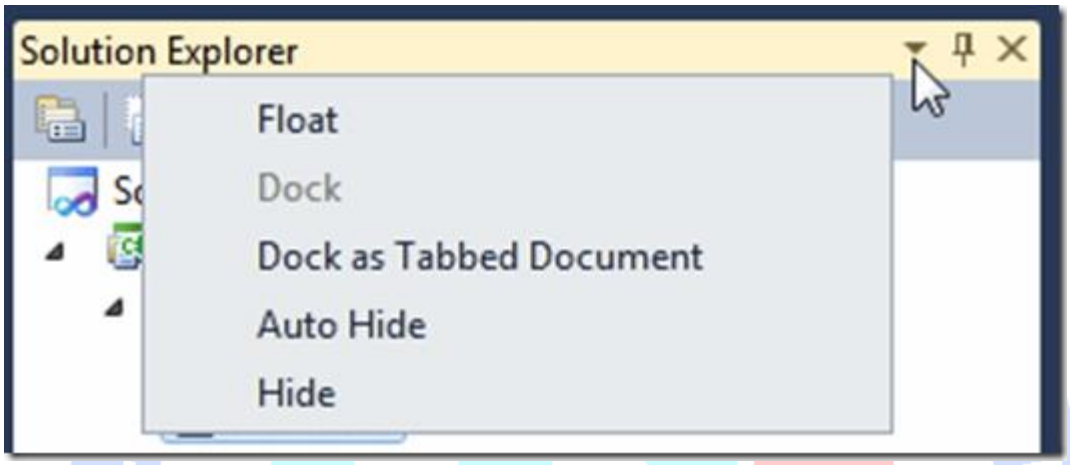
این پنجره از چهار جز شکیل می شود. درست مانند پنجره های دیگر، بخشی دارد به نام نوارتیتر (title bar) که در سمت چپ آن سه گزینه وجود دارد.



با کلیک روی دکمه ی موقعیت پنجره (window position)، فهرستی پدیدار می گردد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

در صورتی که پنجره ی موردنظر به کناره ی صفحه متصل باشد، گزینه ی float فعال می باشد. چنانچه پنجره از قبل در حالت شناور قرار دارد، گزینه ی float فعال نمی باشد. پس از کلیک روی گزینه ی float پنجره ی (متصل به کناره ی صفحه) شناور می شود. پی در پی، می توانید با گرفتن و کشیدن titlebar به وسط صفحه، آن را از حالت متصل (docked) خارج کنید.



به منظور متصل کردن پنجره به کناره ی صفحه، titlebar را بکشید. زیر سرتیتر (titlebar)، فهرستی وجود دارد که به آن نوارابزار (toolbar) می گویند

گزینه ی Properties

گزینه ی show all files؛ تمام فایل های مخفی (hidden) را نمایش می دهد.

دکمه ی refresh، فهرست منابع و فایل ها را بازسازی (refresh) می کند.

دکمه ی view code، برای نمایش کد یک کلاس به کار می رود.

سومین بخش solution explorer بدنه ی اصلی آن است که در آن فایل ها، فولدرها و منابع تشکیل دهنده ی پروژه به نمایش گذاشته می شود. به منظور بزرگنمایی گره (node)، روی اسم آن دوبار کلیک کنید (یا روی دکمه ی آن کلیک کنید). برای کوچک کردن گره، روی آن دوبار کلیک کنید.

ریشه ی (root) لیست، در واقع اسم solution است. زیر root، اسم پروژه ی جاری قرار دارد. در صورتی که، solution دارای چند پروژه باشد، اسم هر یک از پروژه ها زیر solution نمایش داده می شود. درون پروژه ها : فایل ها، فولدرها و منابع وجود دارد. اولین آیتم زیر اسم پروژه References است. پس از گره References (ارجاعات)، با اسم کلاس ها (ی تشکیل دهنده ی پروژه) مواجه می شویم.

چهارمین جز solution explorer تب (tab) می باشد.

Properties Windows

پنجره ی properties، تمام جزئیات فایل ها و منابع سیستم عامل ویندوز که در پروژه به کار برده شده را نمایش می دهد. برای نمایش آن چنانچه، پنجره ی Properties نمایان نباشد، به main menu مراجعه کرده سپس روی View -> Properties window کلیک کنید.

Standard toolbar را باز کرده، گزینه ی Properties را انتخاب کنید.

حال، اگر پنجره ی Properties، در صفحه ی نمایش قابل رویت است، روی تب آن کلیک کنید.

به منظور مشاهده ی ویژگی های سیستم عامل پروژه یا فایل موردنظر(در solution explorer)، روی شی دلخواه کلیک کنید.

با کلیک روی solution، پنجره ی Properties اسم و مسیر (مکان قرارگیری فایل) آن را نمایش می دهد. چنانچه، گزینه ی project را انتخاب کنید، پنجره ی Properties فایل های پروژه ی موردنظر را نشان می دهد

اگر روی فایل کلیک کنید، پنجره ی Properties اسم و مسیر فایل نام برده را به نمایش می گذارد.

پنجره ی Properties بسته به آیتم انتخابی شما (در solution explorer) قسمت های (field) مختلفی را نشان می دهد. می توان برخی تنظیمات را در پنجره ی Properties دستکاری کرد. چنانچه، قسمت (field) ای غیر فعال شود، بدین معنا است که دیگر نمی توان آن را اصلاح کرد.

دسترسی به فایل و باز کردن آن

در برنامه های ویرایش گر متن مثل Notepad، بخشی وجود دارد به نام open file. چناچه سعی دارید فایل c# را با آن باز کنید، ابتدا لازم است که در combo box، نوع فایل را به All files (*) تغییر دهید. سپس، محل قرارگیری فایل را پیدا کرده و روی open کلیک کنید.

چنانچه فایل مذکور عضوی از پروژه جاری است برای باز کردن آن مراحل زیر را دنبال کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

به main menu مراجعه کرده، روی Window و اسم فایل موردنظر کلیک کنید.

در code editor، روی برچسب حاوی اسم فایل کلیک کنید.

در solution explorer، روی اسم فایل دو بار کلیک کنید.

در صورت استفاده از Microsoft Visual C# 2010 Express و Microsoft Visual Studio، برای باز کردن فایل در main menu، روی گزینه های click File -> Open File... کلیک کنید یا به Standard toolbar مراجعه کرده و روی Open File کلیک کنید.

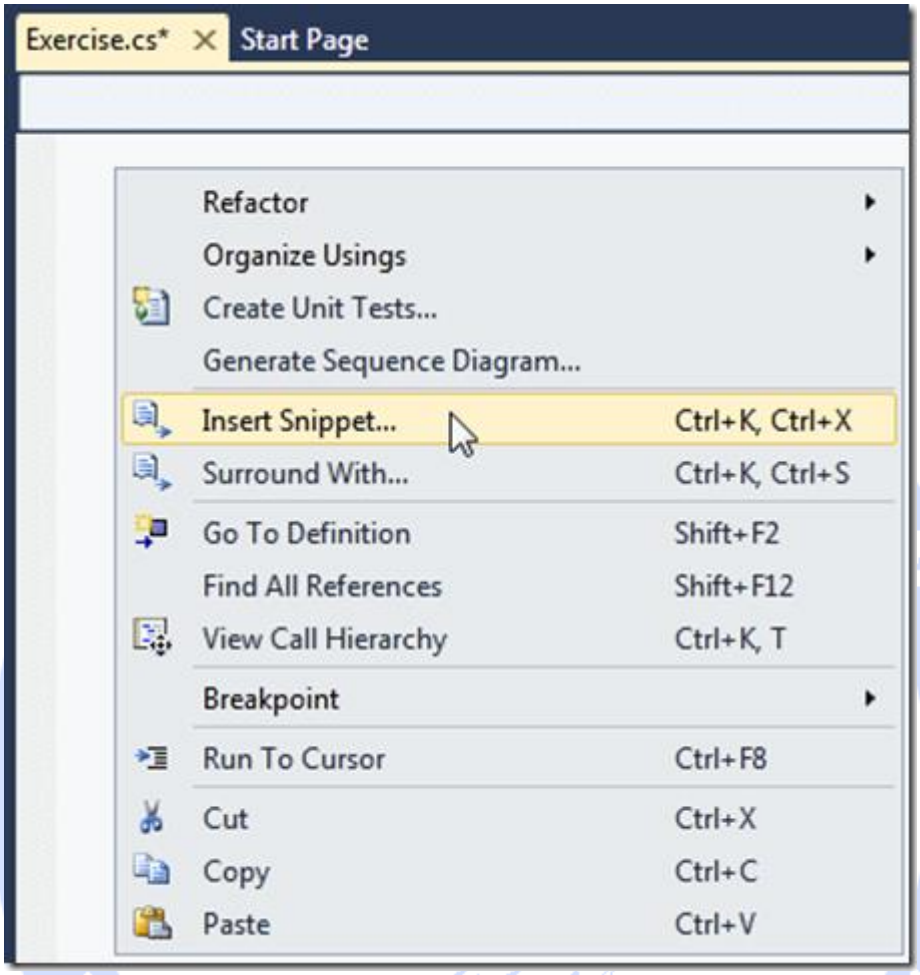
در نتیجه ی هریک از عملیات بالا open file ظاهر می گردد. حال، فایل را از فولدر مرتبط پیدا کرده و روی آن کلیک کنید. اگر visual studio امکانات باز کردن فایل را داشته باشد آن را داخل رابط (interface) خود نمایش می دهد. ولی اگر نتواند آن را باز کند، در کامپیوتر به دنبال برنامه ای که قادر به باز کردن فایل مزبور است می گردد. برای مثال، اگر سعی کنید فایل ویدئویی را با visual studio باز کنید، برنامه ی windows media player به کمک شما می آید و فایل را باز می کند.

مدیریت solution و پروژه

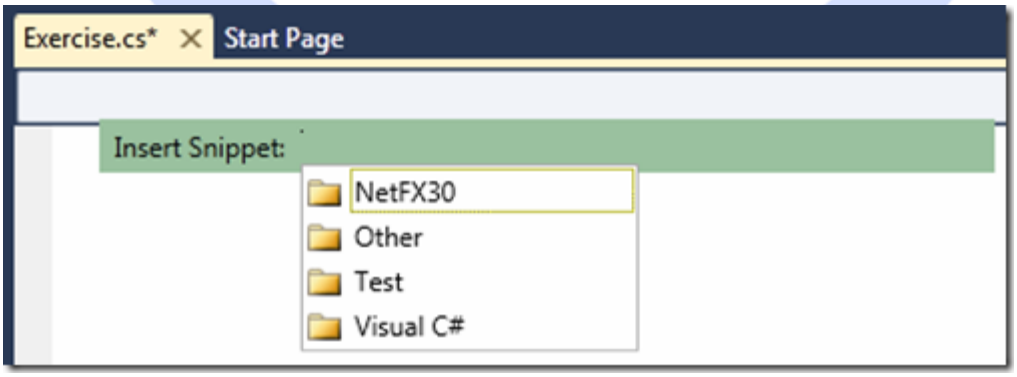
درباره ی نحوه ی ایجاد برنامه ی کاربردی کنسول (console app) مطالبی گفته شد. به کمک visual studio می توان برنامه های کاربردی متفاوتی ایجاد کرد. به همین دلیل، باید ابتدابرای انتخاب گزینه دلخواه پنجره ی محاوره ی New Project را باز کنید.

Code Snippet

visual studio دارای skeleton code های فراوانی است که می توان تنظیم کرد (customize) و به کار برد. کد اصلی (primary code) را برای شما می نویسد و تمام رفتار های پیش فرض مورد نظر را به آن اضافه می کند. پس از این که skeleton code عملیات بالا را انجام داد، می توانید هر بخشی را مایلید حذف کرده یا تغییر دهید. به منظور دسترسی به skeleton code های نام برده، روی قسمتی از فایل که می خواهید آن (skeleton code) را اضافه کنید راست کلیک کرده، سپس گزینه ی insert snippet را انتخاب کنید.

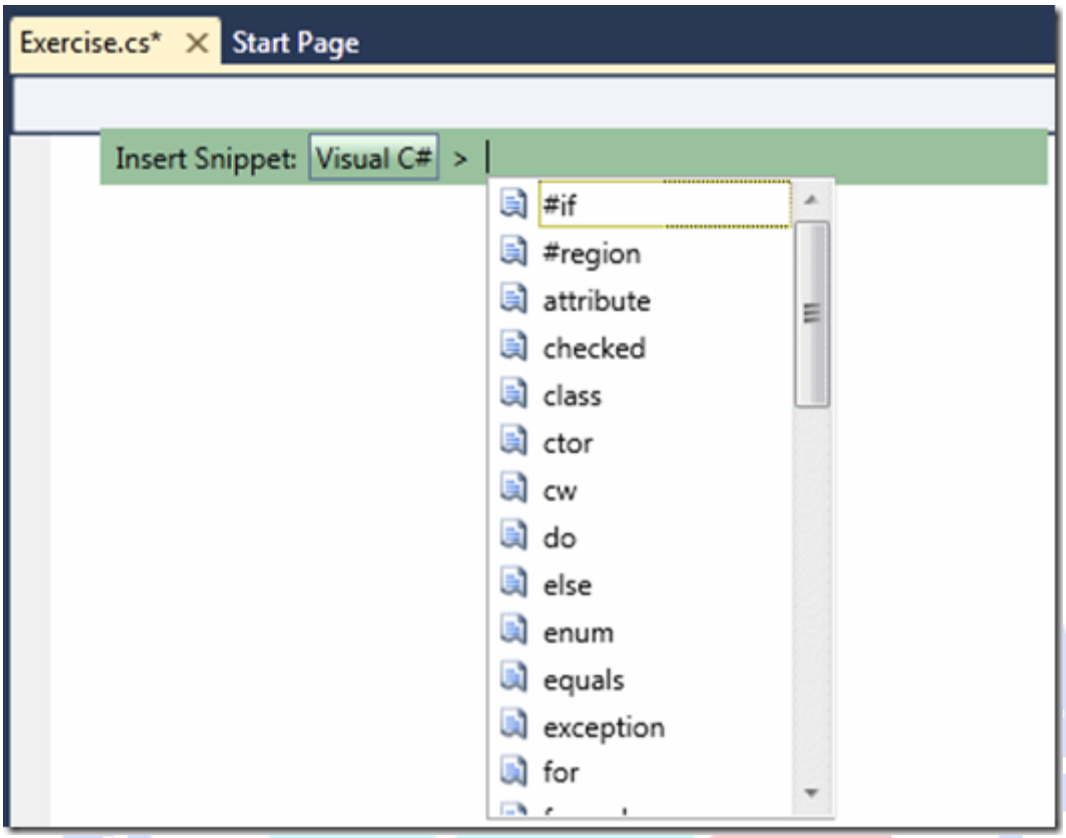


در فهرستی که نمایان می گردد، روی visual c# کلیک کنید.



این کار لیستی از کدهای موجود را به نمایش می گذارد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



روی کد دلخواه دوبار کلیک کنید.

چنانچه، از قبل تعدادی کد نوشته اید و مایلید که آن ها را اصلاح کنید یا کدهای دیگری به آن اضافه کنید، code editor گروهی skeleton code در اختیار شما قرار می دهد. برای این منظور، روی کد دلخواه (کدی که می خواهید اصلاح کنید) راست کلیک کرده، سپس گزینه ی surround with را انتخاب کنید. اکنون، لیستی پدیدار می شود که گزینه ی موردنظر شما در آن درج شده.

رنگ کدها

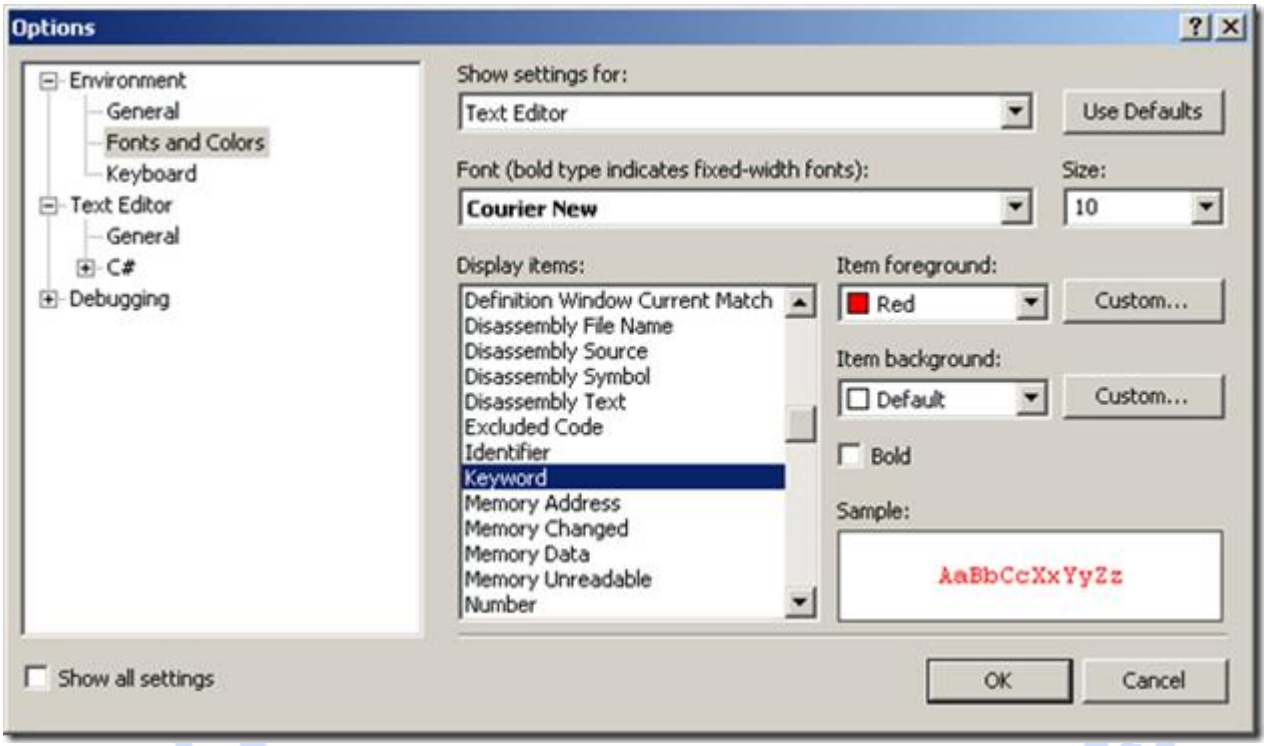
کدها به صورت اساسی در محیطی گسترده با پس زمینه ای سفید نوشته می شوند. در این محیط است که شما با استفاده از صفحه کلید، کد موردنظر را با کاراترهای خوانا و معمول درج می کنید. code editor به منظور تمیز دادن خط ها و کلمات متن از رنگ های مختلف استفاده می کند.

می توان این رنگ ها را تنظیم کرد. برای این منظور، main menu را باز کرده و گزینه ی Tools -> Options.. را انتخاب کنید. در پنجره ی options، قسمت Environment، روی گزینه ی fonts and colors کلیک کنید. برای تنظیم رنگ دسته (category) ی موردنظر، در قسمت

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

Display Items، دسته ی دلخواه را انتخاب کنید. حال، می توانید رنگ انتخابی را در لیست پایین افتادنی item foreground تنظیم کنید.

چنانچه مایلید پس زمینه ی کلمات دسته ی مورد نظر رنگی شود، به قسمت item background مراجعه کنید.



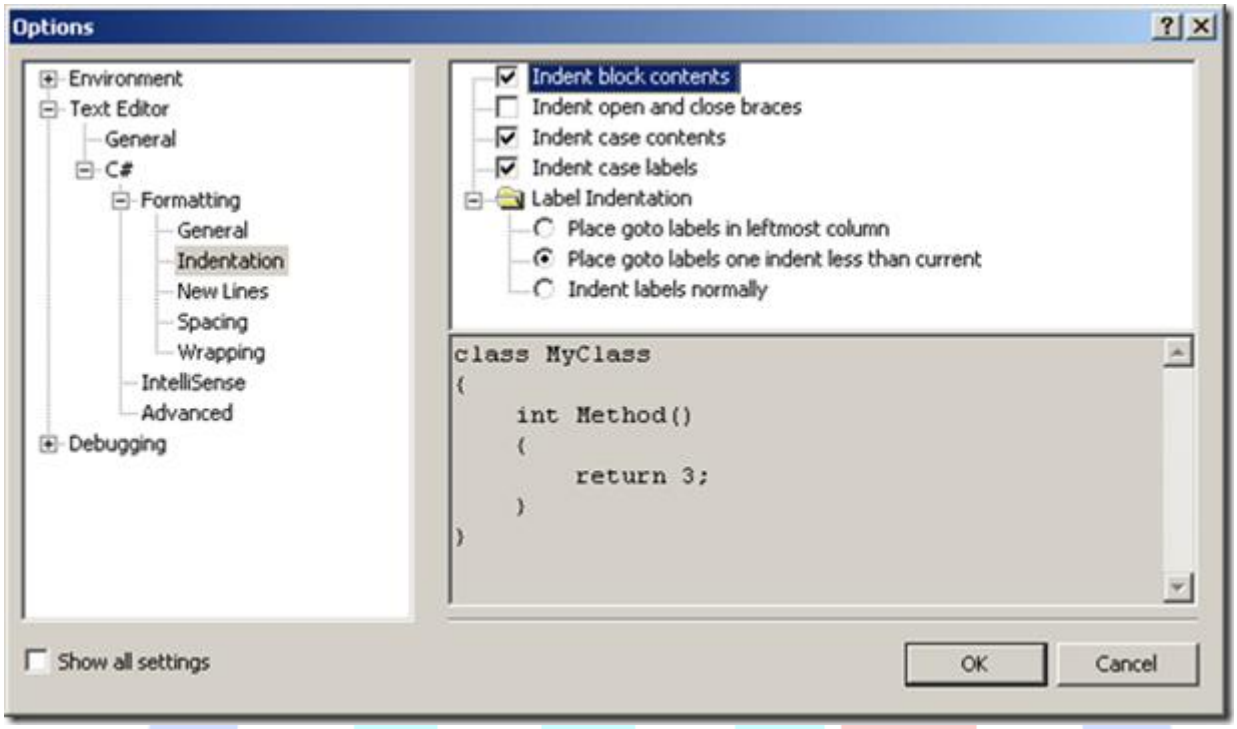
در هر دومورد، فهرست ثابتی از رنگ های معین پدیدار می شود. اگر مایلید رنگ جدیدی به این لیست اضافه شود، روی دکمه ی custom کلیک کنید. حال، پنجره ی محاوره color نمایش داده می شود که می توان در آن رنگ دلخواه را به وجود آورده و اضافه کرد.

توگذاری (Indentation)

ویژه گی دیگر که به خوانایی هر چه بیشتر برنامه ی شما کمک می کند توگذاری (indentation) است. indentation در واقع به گروه بندی خط های کد مورد نظر بر مبنای دسته ی مرتبط گفته می شود. به منظور تعیین حد و مرز (delimit) کدها، باید آن ها را به اندازه ی دو جای خالی یا یک تَب (tab) تو بگذارید. توگذاری (indentation) باید مراتب افزایشی داشته باشد. به عبارت دیگر، خطی که زیر مجموعه ی خط دیگری است، باید بیشتر تو گذاشته شود.

برای مدیریت indentation کد، main menu را باز کرده و روی گزینه ی Tools -> Options... کلیک کنید. در لیست سمت چپ، C# را باز کرده و بعد از بزرگ نمایی گزینه ی Formatting، روی indentation کلیک کنید. حال، می توان گزینه های سمت راست را تنظیم کرد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



پس از ایجاد تغییرات مورد نظر، روی گزینه ی ok کلیک کنید.

ذخیره سازی پروژه

اگر برنامه ی دلخواه خود را با text editor (ویرایش گر متن) ایجاد می کنید، باید فایل های آن را در فولدری ذخیره کنید. در نسخه های قبلی visual studio (۲۰۰۳ و ۲۰۰۲)، کاربر مجبور بود خود به صورت رسمی یک پروژه ایجاد کند و بعد آن را ذخیره کند. بعد ها مایکروسافت دریافت که خیلی از پروژه هایی که برنامه نویسان به وجود می اورند، جنبه ی آزمایشی دارد. بنابراین، امکان جدیدی به برنامه اضافه کرد که به کاربر اجازه می دهد یک پروژه به صورت موقت ایجاد کند و تصمیم بگیرد که آن را save کند یا نه. این کار، پروژه را در رسانه ی داده ذخیره می کند و کاربر می تواند بعد ه ها به آن مراجعه کرده و از آن استفاده کند.

پس از طی کردن تمام مراحل نصب visual studio 2010، در Documents فولدری به نام Visual Studio 2010 به وجود می آید. فولدر doctments را با نام های دایرکتوری شخصی (personal derictory) یا personal drive نیز می شناسند. در فولدر Visual Studio 2010، subfolder دیگری به projects به وجود می آید. در این دایرکتوری است که پروژه های شما به صورت پیش فرض ذخیره می شوند.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

به منظور ذخیره سازی پروژه، به Standard toolbar مراجعه کرده، و روی دکمه ی  Save all کلیک کنید. پی در پی، می توانید main menu را باز کرده و گزینه ی File -> Save All را انتخاب کنید. چناچه، پروژه، قبلاً ذخیره شده ولی می خواهید آن را تحت اسم دیگری ذخیره کنید باید این دستور را دنبال کنید : File -> Save project name As…

بازکردن پروژه

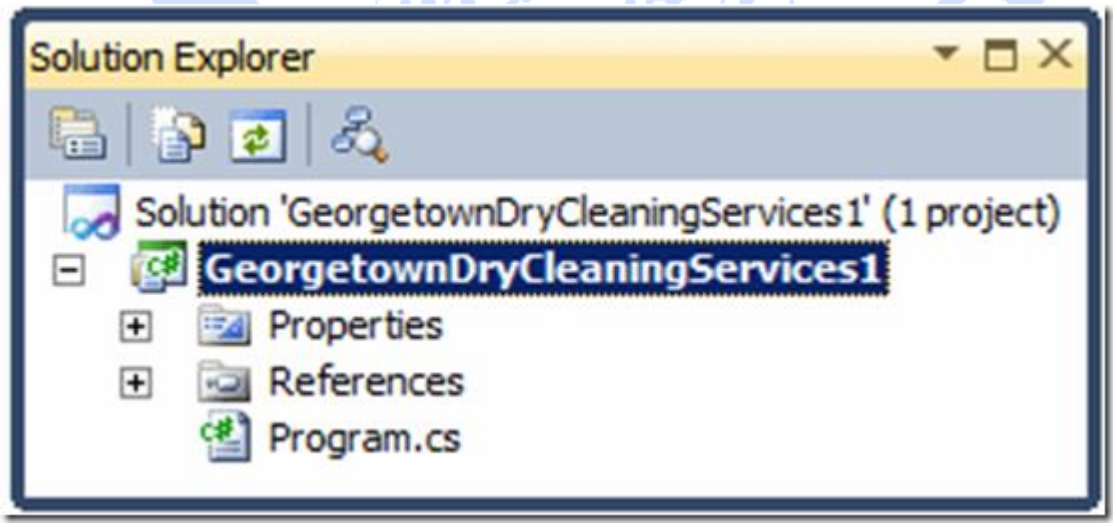
شیوه های مختلف بازکردن پروژه، در زیر فهرست شده

- Main menu ->File-> Open project
- Start -> Open project
- Ctrl + Shift + O

Solution

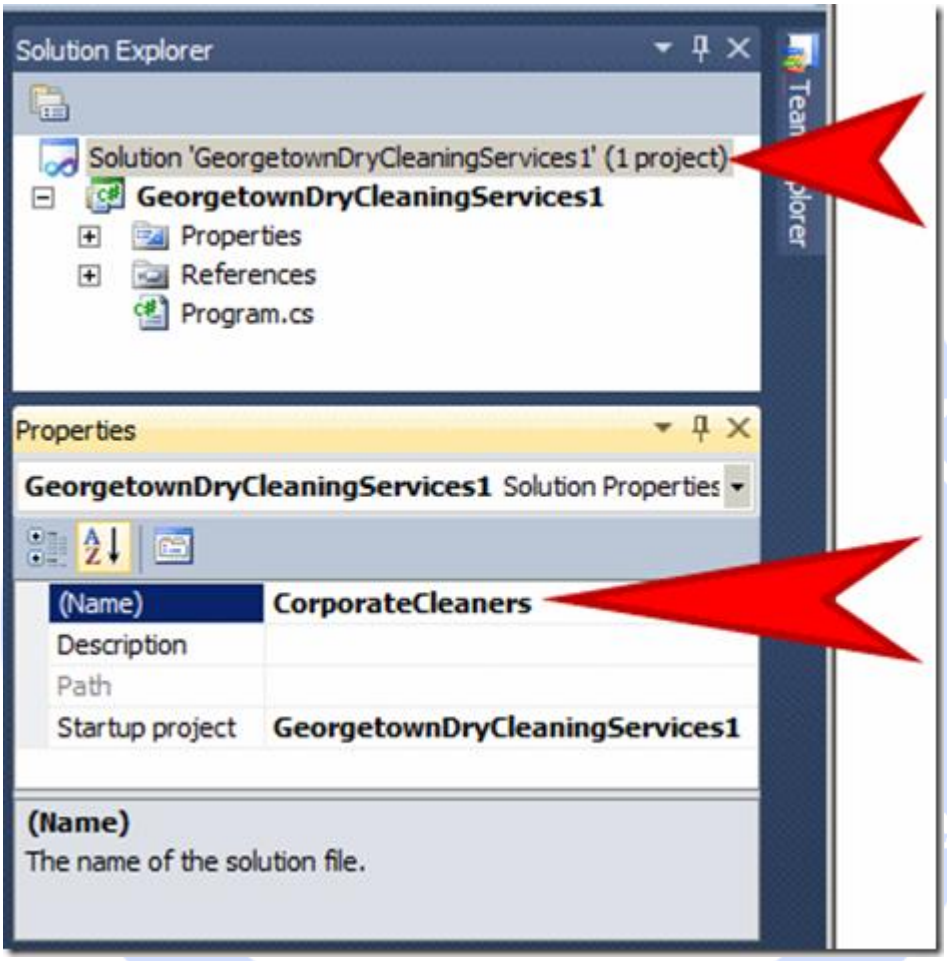
Solution به منظور هماهنگ سازی (ایجاد هماهنگی بین) بخش/جنبه های مختلف یک برنامه (app) به کار می رود. هر پروژه، تنها یک بخش از برنامه ی کاربردی مورد نظر را تشکیل می دهد. علاوه بر کدی که می نویسید، لازم است آیتم های دیگری نیز به برنامه اضافه کنید.

توجه داشته باشید که اسم پروژه و solution یکسان می باشد. می توانید اسامی آن ها را در solution explorer مشاهده کنید.



آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

Solution و پروژه ممکن است اسامی متفاوت داشته باشند. به منظور تغییر اسم solution، (در solution explorer)، روی اولین گره (node) کلیک کنید. سپس، (در پنجره ی Properties)، روی Name کلیک کرده و اسم دلخواه را وارد کنید.

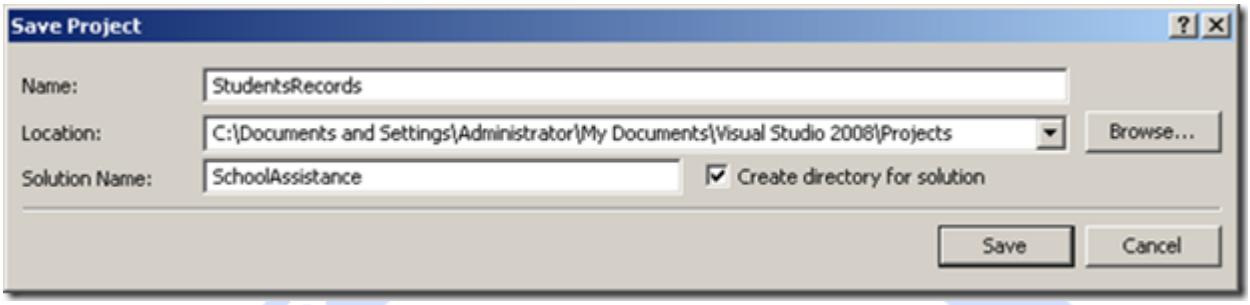


لازم به ذکر است که اسم مزبور به طور کامل موقتی است. اگر می خواهید solution را به صورت دائم ذخیره کنید، دو روش پیش روی شما وجود دارد.

در صورتی که solution را برای اولین بار ذخیره می کنید، پنجره ی Save Project پدیدار می گردد. به صورت پیش فرض، Microsoft Visual Studio، دایرکتوری شخصی شما را به عنوان مسیر ذخیره سازی فایل انتخاب می کند. به این مسیر location می گویند. در location نام برده، Microsoft Visual Studio فولدری به وجود می آورد به عنوان solution پروژه مورد نظر. حال، solution باید در فولدر مختص به خود قرار گیرد. همان طور که پیش توضیح دادیم، Microsoft Visual Studio از اسم پروژه برای solution نیز استفاده می

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

کند.برای جلوگیری از این رخداد، می توانید رشته (string) را در کادر متن Solution Name (textbox) تغییر دهید. به خاطر داشته باشید که اسم دلخواه را باید در کادرمتن Name وارد کنید. به مثال زیر توجه کنید.



زمانی که (برای اولین) پروژه ای را ذخیره می کنید، Microsoft Visual C# به صورت پیش فرض فولدری برای آن در مسیر My Documents\Visual Studio 2008\Projects folder به وجود می آورد. هم چنین از اسم solution برای نام گذاری فولدر استفاده می کند. تعدادی فایل جدید ایجاد کرده و آن ها را در فولدر تازه به وجود آمده ذخیره می کند. سپس، داخل فولدر solution، subfolder جدیدی با نام پروژه ی مورد نظر ایجاد می کند. جدا از subfolder (ای که اسم آن Project است)، فولدر دیگری به نام debug به وجود می آید. همچنین فولدر دیگری به اسم Debug، در subfolder اسم پروژه به وجود می آورد.

چناچه، پروژه ی مورد نظر از قبل ذخیره شده باشد، ولی مایلید که اسم solution را عوض کنید، به main menu مراجعه کرده و روی File - > Save solution-name.sln As.. کلیک کنید. این کار منجر به بالا آمدن کادر Save as می شود که در آن اسم solution آماده ی تغییر و ذخیره سازی است.

ساختن پروژه

پس از ایجاد پروژه و نوشتن کد، برای مشاهده ی نتیجه، ابتدا باید به ساختن برنامه (app) بپردازید. این کار یک فایل اجرایی (executable) خلق می کند.

زبان C#، به منظور برنامه نویسی (ساختن برنامه ی جدید) امکانی به نام compiler (مترجم) را در اختیار کاربر قرار می دهد. compiler یک برنامه ی کامپیوتری است که خود از زیربرنامه (subprogram) های داخلی متعدد دیگری تشکیل شده. اولین و مهم ترین این زیربرنامه ها (parser تجزیه کننده) خوانده می شود. وظیفه ی parser اسکن یا پوییدن فایلی است که (خود بخشی از) برنامه را دربردارد. دستور نحوی (syntax)، کلیدواژه ها و تمامی کلمات ناشناس توسط parser بررسی می شوند. اگر parser در جستجو ی خود مشکل یا خطایی را پیدا کند، یا همان جا متوقف می شود و یا لیستی از تمام مشکلاتی که به آن ها برخورده تهیه می کند، سپس لیست مذکور را برای شما به

نمایش می گذارد. گاهی تنها به خطی اشاره می کند (تنها شما را به خطی ارجاع می دهد) که مشکل در آن یافت شده و گاهی نیز به خطی ارجاع می دهد که مشکل اثر خودش را در آن جا نشان داده ولی خود مشکل در جای دیگری یافت می شود. البته، نحوه ی حل مشکلات فوق را با تجربه یاد می گیرد.

اگر parser در اسکن خود با مشکلی مواجه نشد (یا پس از اینکه مشکل مزبور را برطرف کردید)، برنامه ی نام برده (parser) نتایج بدست آمده را به compiler ارسال می کند. حال، compiler برنامه ی دیگری به نام linker را فرا می خواند. linker کلیه ی فایل های برنامه را بررسی می کند (چه یک فایل داشته باشد چه چند فایل). سپس، linker تعدادی از فایل های ارسال شده توسط C# compiler (تنها آن دسته فایل هایی که برنامه برای فعالیت، نیاز مبرم به آن ها دارد و نه کلیه ی فایل ها یی که .Net Framework فراهم می کند) را با فایل های شما ادغام کرده تا دستورات شما را درست دریافت کند و نتایج دلخواه شما را ارائه دهد. در صورتی که مشکل جدی ای وجود نداشته باشد، compiler برنامه ی مورد نظر را می سازد. به خاطر داشته باشید که این بدین معنا نیست که اصلاً هیچ مشکلی وجود ندارد، بلکه compiler مشکلی پیدا نکرده و هنوز احتمال این وجود دارد که نتیجه ی حاصل مورد دلخواه شما نباشد.

باید در نظر داشته باشید که تمام برنامه های بالا (linker، parser، debugger) همه گی زیر مجموعه ی برنامه ی بزرگتری به نام compiler هستند. از این به بعد، از compiler به عنوان برنامه ای یاد می شود که زبان (انگلیسی) را به زبان کامپیوتر ترجمه می کند.

Compiler ای که برنامه ی Microsoft.NET Framework ارائه می دهد csc گفته می شود. مشابه خیلی از برنامه های دیگر، compilerپسوند .exe را دارد. Csc اسم استانداردی برای compiler نمی باشد. به این معنا که C# compiler اسم دیگری دارد : csc.exe اسم compiler ای است که ما استفاده می کنیم.

به منظور نوشتن برنامه از طریق پنجره ی فرمان (command prompt)، از csc.exe compiler استفاده می کنیم. پس از دانلود .NET Framework از سایت رسمی مایکروسافت، باید آن را روی رایانه ی خود نصب کنید. برنامه ی مورد نظر ما csc.exe است. مسیر نصب این برنامه به صورت پیش فرض C:\Windows\Microsoft.NET\Framework\v4.0.21006 می باشد.

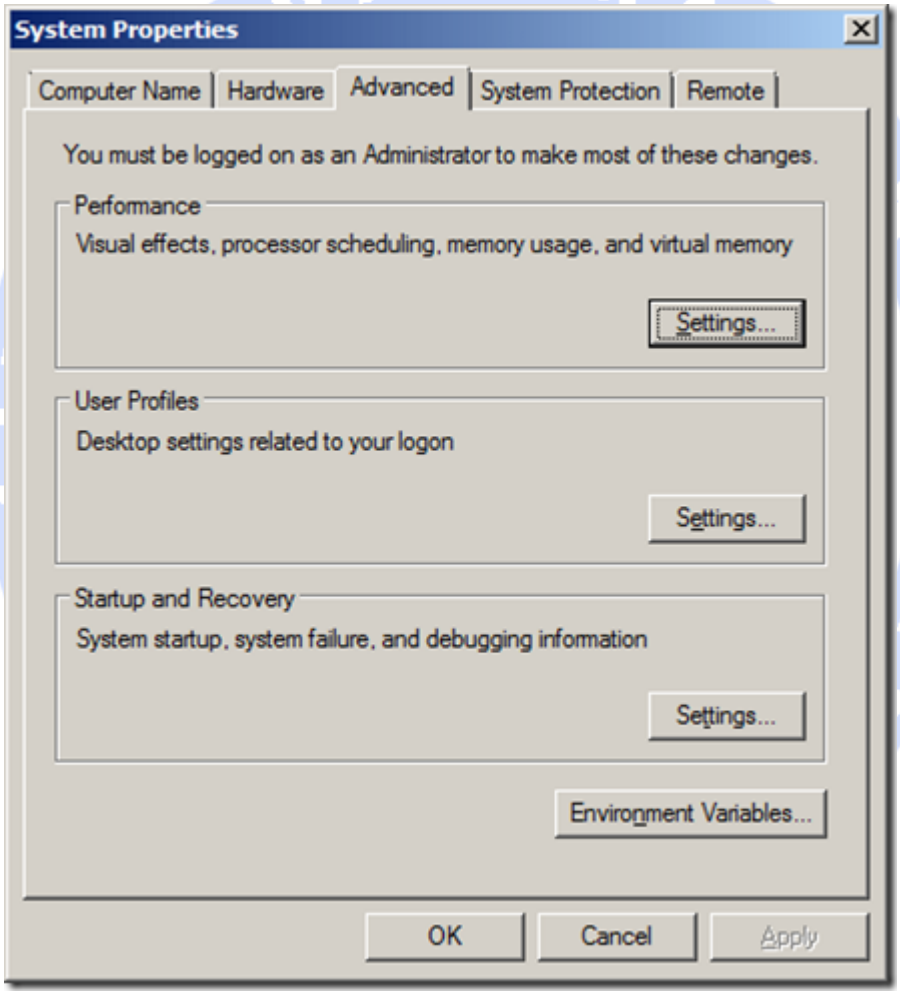
باید مسیر csc.exe را به مسیر Environment Variables اضافه کنید. برای شروع،می توان از ابزاری همچون windows explorer کمک گرفته و فولدر مورد نظر را در جایی که csc نصب شده نمایش دهید.

مسیر مورد نظر را از بالاترین combo box انتخاب کرده و آن را در قسمت clipboard کپی کنید. پنجره ی control panel را باز کرده و روی آیکون system and security کلیک کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

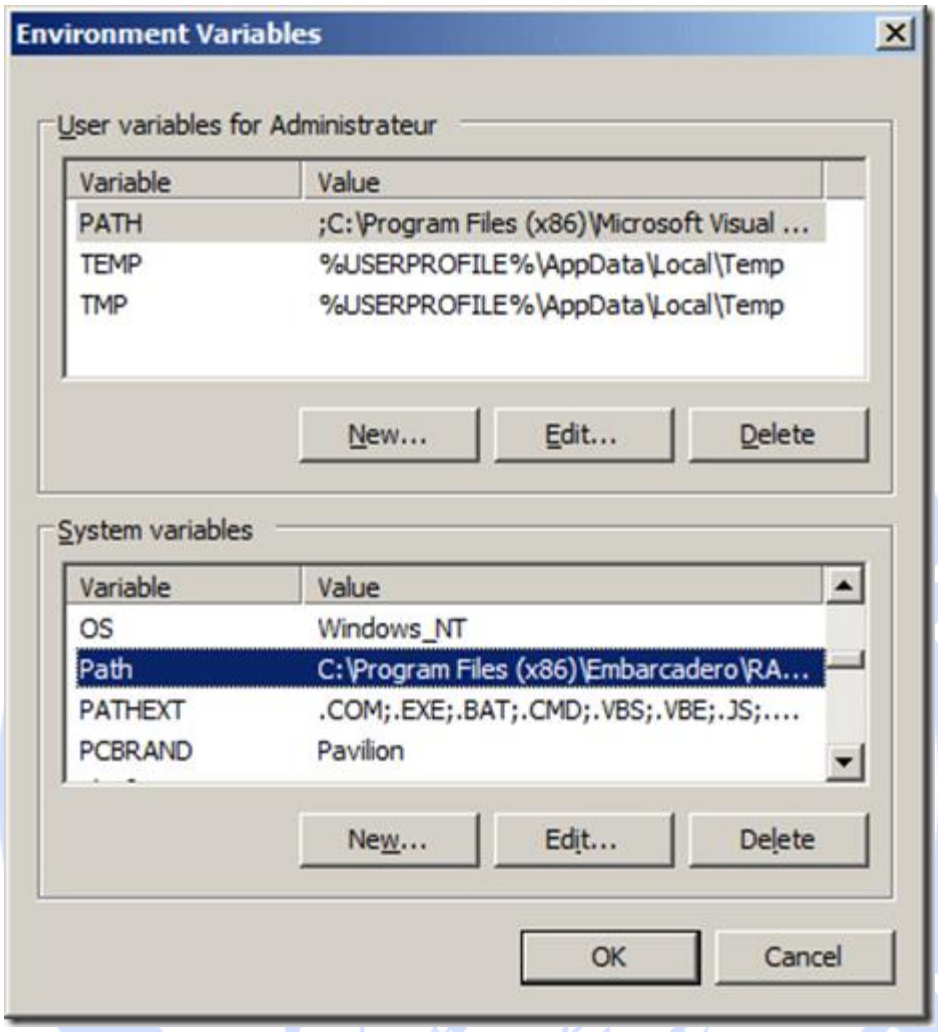
حال، در قسمت system and security، روی گزینه ی system کلیک کنید.

پس از نمایان شدن پنجره ی system، گزینه ی change setting را انتخاب کنید. اکنون، تب Advanced را باز کرده و روی Environment Variables کلیک کنید.



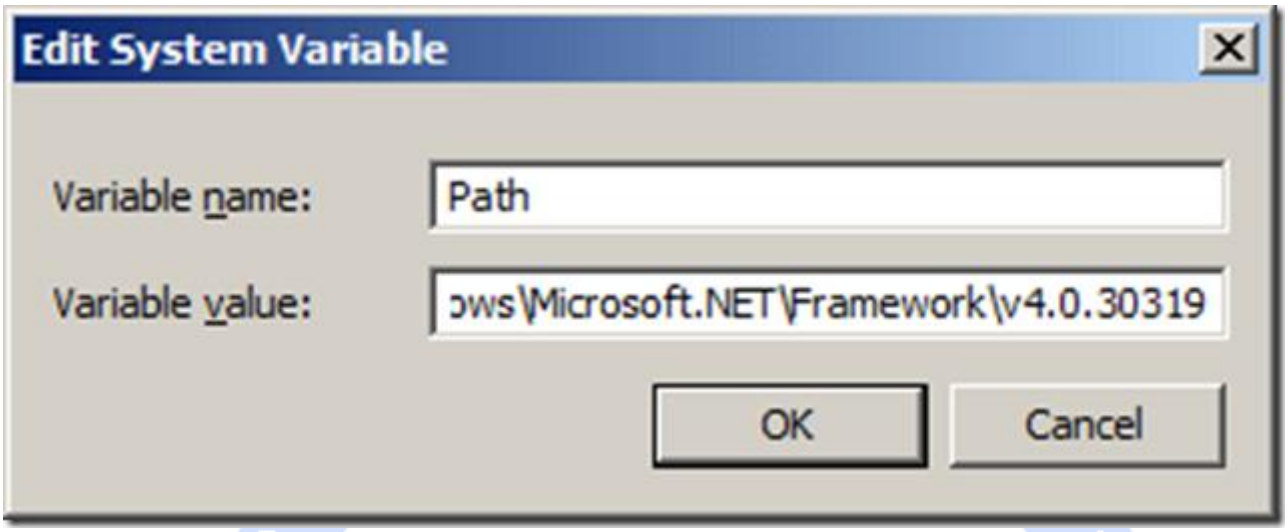
در بخش System Variables، روی Path دو بار کلیک کنید، یا یک بار روی آن کلیک کرده سپس Edit را انتخاب کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



دکمه ی End را فشار داده، علامت ";" را تایپ کنید. حال، مقدار (value) ای را که از clipboard کپی کرده بودید را در این قسمت پیاده کنید (paste).

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



سه بار روی ok کلیک کنید.

پنجره ی فرمان (command prompt) را باز کنید. حال، برای وارد شدن به درایو اصلی (root drive)، CD\ را تایپ کرده و دکمه ی Enter را فشار دهید. CD\ را تایپ کرده و دکمه ی space را فشار دهید و به دنبال آن فولدر (یا subfolder) هایی که فایل در آن قرار دارد را در این قسمت درج کنید. سپس دکمه ی Enter را فشار دهید. برای ترجمه (compile)، csc را به دنبال اسم فایل و پسوندش (با تایپ کردن) وارد کنید.

فایلی که در نتیجه ی این عملیات تولید می شود پسوند .exe را دارد. به صورت پیش فرض، اسم این فایل با فایلی که ابتدای کار استفاده کردید یکی است. برای دریافت برنامه ی اجرایی (executable) ای با نام دلخواه، پس از کاراکتر csc، /out: را تایپ کرده و به دنبال آن اسم دلخواه، به علاوه ی پسوند .exe، فاصله (space)، اسم فایلی که ابتدای کار ایجاد کرده بودید به همراه پسوند آن را وارد کنید. فرمول بالا به این شکل است: csc /out:NameOfExecutate.exe Filename.cs

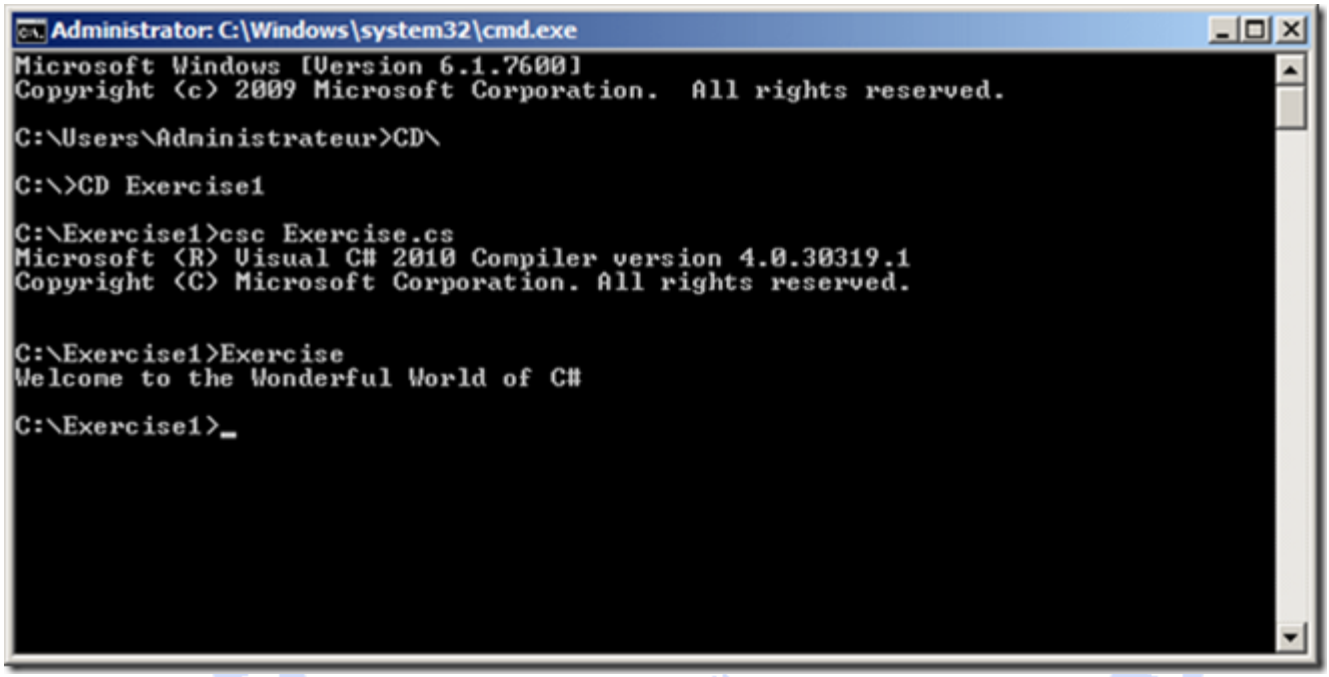
NameOfExecutate نشانگر اسم دلخواه executable شما می باشد.چنانچه اسم مورد نظر تنها یک کلمه است، می توانید به راحتی آن را تایپ کنید. ولی اگر اسم متشکل از چند کلمه است، باید آن را داخل علامت " قرار دهید.

اگر برای تولید برنامه ی مورد نظر از text editor استفاده می کنید و تعداد فایل هایی که ایجاد کرده اید زیاد است، باید هنگام انجام فرایند ترجمه (compiling)، به تک تک فایل ها ارجاع دهید (reference). برای این منظور، لازم است که اسم هر فایل را به همراه پسوند مربوطه ی آن به بخش پایانی اضافه کنید: cscFileName1.cs FileName2.cs FileName_n.cs

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

اجرای پروژه

پس از ساختن پروژه، شما و کاربرانتان می توانید آن را اجرا کنید. به منظور اجرای پروژه، (از طریق command prompt)، اسم فایل را به همراه پسوند .exe با Enter وارد کنید.



در صورتی که با visual studio کار می کنید، برای اجرای برنامه ، به main menu مراجعه کرده و روی گزینه های Debug -> Start Debugging کلیک کنید.

اجرای برنامه

به منظور اجرای برنامه، main menu را باز کرده و گزینه ی Debug -> Start Debugging را انتخاب کنید.

پس از مشاهده ی نتیجه در پنجره ی DOS، با زدن دکمه ی Enter آن را ببندید.

برنامه ی visual studio را نیز ببندید.

در پاسخ به در خواست : آیا مایلید save کنید، No را بزنید.

متغییرها (Variables)

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

مبنع ذخیره سازی

کامپیوتر یک وسیله ی الکترونیکی است که به حل مشکل خاصی می پردازد یا وظیفه ی مشخصی را انجام می دهد. برای مثال، یک دستگاه تناسب اندام به شخص کمک می کند، هیکل خود را رو فرم نگه دارد یا یک دوربین دیجیتال از چیزهای مختلف عکس می گیرد.

وسایل الکترونیکی به منظور حل مسائل عمومی نیز به کار می روند. برای مثال، از رایانه ی شخصی برای انجام کارهای عمومی مثل محاسبات، پردازش کلمه، یا ذخیره سازی بانک های اطلاعاتی استفاده می شود.

برای انجام وظایف مختلف، کامپیوتر باید مقدارهای (value) مشخصی دریافت کند. این کار ممکن است توسط شخصی صورت بگیرد که دستوری را از صفحه کلید تایپ می کند (به عنوان مثال از دستگاه تناسب اندام، تلفن همراه، یا کامپیوتر شخصی). در برخی از موارد، مقادیر موردنظر (از داخل) الکترونیکی و از منابع مختلف به کامپیوتر داده شود.

به منظور مدیریت این ارتباطات، کامپیوتر از صفحه ی تختی به نام motherboard استفاده می کند. خیلی از بخش های کامپیوتر به این تخته ی اصلی متصل هستند و از آن دستور دریافت می کنند. بخش دیگری وجود دارد که وظیفه ی اصلی آن پردازش و انجام محاسبات است که به آن پردازش گر می گویند و به تخته ی اصلی (motherboard) وصل می باشد.

حال، مقادیری که کامپیوتر دریافت می کند باید در قسمتی به نام mem یا y یا حافظه ذخیره شود. رایانه از دو نوع حافظه برای این منظور استفاده می کند. حافظه ی موقت و حافظه ی پایدار. از حافظه ی موقت برای ذخیره کردن اطلاعاتی استفاده می شود که حالت موقتی داشته و پس از گذشت زمان مشخصی پاک می شوند. برای مثال، حافظه ی نام برده اطلاعات را هنگامی که رایانه روشن است در خود حفظ می کند و آن را تا زمانی که کامپیوتر روشن است نگه می دارد ولی به محض خاموش شدن رایانه اطلاعات مزبور پاک می شوند.

حافظه ای که اطلاعات و مقادیر به صورت موقت در آن ذخیره می شود random access mem یا y یا RAM نامیده می شود.

فرض کنید، حافظه ی کامپیوتر یک سینی کیک است متشکل از چند بخش، که هریک حامل چیزی است.

توجه داشته باشید که حافظه بزرگ تر است یک کیک است و از میلیون ها جای خالی تشکیل شده.برنامه نویس مدام به compiler دستور می دهد که مقادیر را به صورت موقتی در RAM ذخیره کند. اگرچه اندازه ی حافظه ی موقت چندان بزرگ نیست، باید به خاطر داشته باشید جای زیادی برای ذخیره ی اطلاعات لازم دارد. در واقع، این تنها برنامه ی شما نیست که از RAM استفاده می کند. برای مثال، هنگامی که رایانه ی خود را راه اندازی می کنید، سیستم عامل (windows) و دیگر برنامه ها ی آن RAM را اشغال می کنند.

هنگامی که برنامه ای را اجرا می کنید، compiler بخشی از RAM را به آن برنامه اختصاص می دهد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

به این خاطر که برنامه های زیادی از RAM استفاده می کنند، به منظور ذخیره سازی مقادیر در آن، باید اطلاعاتی در اختیار آن قرار دهید. باید مقدار حافظه ای را که به آن نیاز دارید مشخص کرده و اسم معینی برای آن قسمت خاص از حافظه که value ها در آن ذخیره می شود انتخاب کنید. به ترکیبی از این اطلاعات متغییر (variable) گفته می شود.

معرفی متغییرها

۱. برنامه ی Microsoft Visual C# 2010 Express یا Microsoft Visual Studio 2010 را اجرا کنید.

۲. به منظور ایجاد برنامه ی کاربردی (app) جدید، در Start Page روی گزینه ی New Project کلیک کنید.

۳. در فهرست میانی، گزینه ی Empty Project را انتخاب کنید.

۴. اسم موردنظر را به gdcs2 (ge) یا ge town cleaning services تغییر دهید.

۵. روی ok کلیک کنید.

۶. به نظور ایجاد فایل ویژه ی کد مورد نظر، به main menu مراجعه کرده و project را انتخاب کنید.

۷. از لیست میانی code file را انتخاب کنید.

۸. اسم نام برده را به cleaning یا der تغییر دهید.

۹. روی گزینه ی Add کلیک کنید.

۱۰. در داکیومنت خالی دستورات زیر را تایپ کنید.

```
class Order
{
  static void Main(string[] args)
  {
    System.Console.WriteLine("Georgetown Dry Cleaning Services");
    System.Console.ReadKey();
  }
}
```

نام گذاری متغییر ها

متغییرها باید نام مشخصی داشته باشند و برای ایجاد نام برای آن ها باید از قوانین خاصی پیروی کرد. کلماتی وجود دارند که از آن ها نباید تحت هیچ شرایطی به عنوان نام متغییر استفاده کرد زیرا خود برنامه کلیدواژهای مزبور را به کار می برد. این کلیدواژه ها در زیر فهرست شده.

abstract	continue	finally	interface	out (generic)	short	typeof
as	decimal	fixed	internal	out (methods)	sizeof	uint
base	default	float	is	override	stackalloc	ulong
bool	delegate	for	lock	params	static	unchecked
break	do	foreach	long	private	string	unsafe
byte	double	goto	namespace	protected	struct	ushort
case	else	if	new (generic)	public	switch	using
catch	enum	implicit	new (LINQ)	readonly	this	virtual
char	event	in (foreach)	new (variable)	ref	throw	void
checked	explicit	in (generic)	null	return	true	volatile
class	extern	int	object	sbyte	try	while
const	false		operator	sealed		

واژه های دیگری هستند که با وجود این که جز کلیدواژه های برنامه ی C# نیستند، استفاده از آن ها به شما توصیه نمی شود. به این خاطر که به کار بردن آن ها ممکن است منجر به بروز اختلال (conflict) در کد شود. از واژه های داده شده به عنوان کلیدواژه های متنی (contextual keyw یا ds) یاد می شود.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

add	global	let	partial (type)	set	where (generic)
dynamic	group	orderby	remove	value	where (query)
from	into	partial (method)	select	var	yield
get	join				

همان طور که پیش تر ذکر شد، اسم گذاری برای هر چیزی در برنامه نویسی قوانین خاص خود را دارد. قوانین استانداردی هست که توسط C# تعریف می شوند ولی شما می توانید قوانینی را بر مبنای سلیقه ی خود نیز به وجود بیاورید. قوانینی که برای تعیین اسم متغییر باید پیروی کنید به شرح زیر می باشد.

نام متغییر می تواند تنها از یک حرف تشکیل شود(A، B، C، D، E، F، G، H، I، J، K، L، M، N، O، P، Q، R، S، T، U، V، W، X ، Y، Z، a، b، c، d، e، f، g، h، i، j، k، l، m، n، o، p، q، r، s، t، u، v، w، x، y، z یا) . این حروف متعلق به الفبای انگلیسی آمریکایی می باشد، ولی زبان C# انگلیسی بین المللی را نیز می پذیرد.

در C#، نام ممکن است از تنها یک علامت (_) ساخته شود. با این وجود، توصیه می شود از این کار خودداری کنید زیرا خواندن کدی که از تنها یک _ تشکیل شده آسان نیست.

از به کار بردن این علامت ها در نام متغییر خودداری کنید. !، ' "، / $، % ?، & *، ( )، + #، \ @، < >، [ ]، ; {، } {

در صورتی که نامی متشکل از بیش ازیک کاراکتر باشد، باید با حرف یا _ شروع شود.

پس از آغاز کاراکتر با _ یا حرف، نام می تواند از ترکیبی از حروف، ارقام (۰، ۱، ۲، ۳، ٤، ٥، ٦، ۷، ۸، یا9)، و / یا _ تشکیل شود.

در نام نمی توان از فاصله (space) استفاده کرد.

علاوه بر قوانین بالا، برنامه نویس می تواند قوانین خود را تعریف کند. ولی در تعریف همین قوانین باید محدودیت های ذکر شده در نظر گرفته شود.

اگر نامی در تنها یک کلمه خلاصه شود، اکثر برنامه نویسان از حروف کوچک استفاده می کنند.

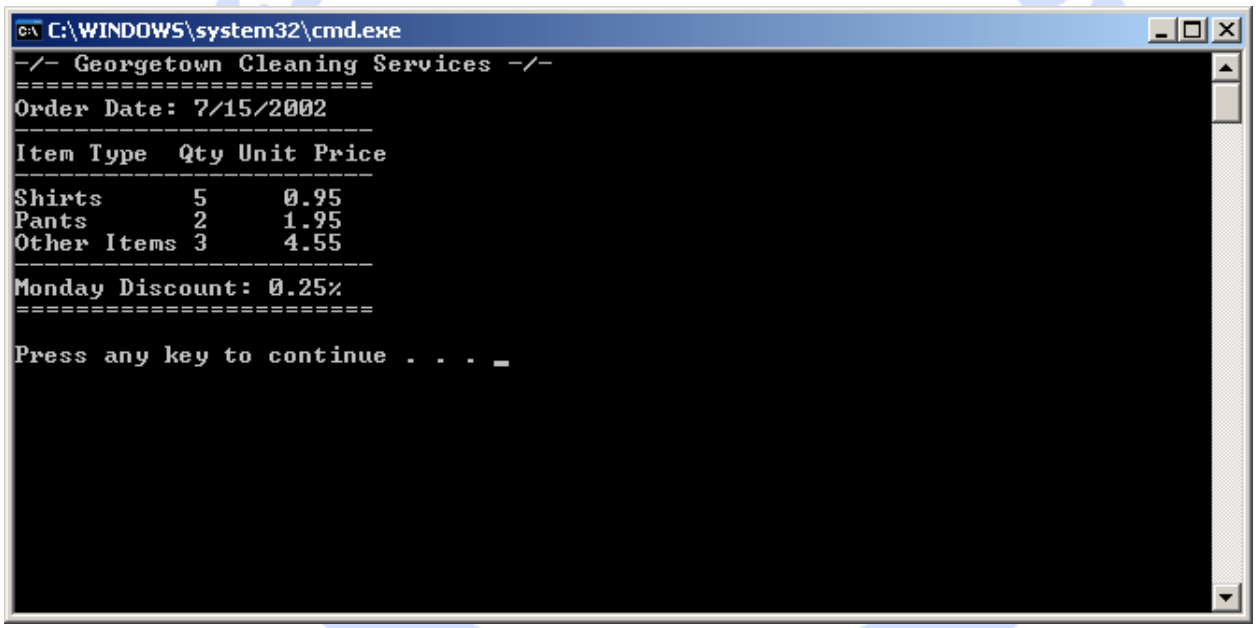
آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

زمانی که نام متغییر متشکل از چند کلمه است، بیشتر برنامه نویسان از camel notation استفاده می کنند که در آن اولین کلمه با حروف کوچک نوشته می شود و اولین حرف کلمات بعدی همگی با حروف بزرگ نوشته می شوند.

زبان C# به کوچک بزرگی حروف حساس است، یعنی کلمات case، Case و CASE با هم کامل متفاوت اند. برای مثال، main همیشه Main نوشته می شود.

مقدارها و متغییرها در کنسول

برنامه های زبان C# نتایج خود را در پنجره ی DOS به نمایش می گذارند. به مثال زیر توجه کنید.



برای نشان دادن مقدار معینی در پنجره ی فوق، می توانید آن را داخل پرانتز System.Console.Write() یا System.Console.WriteLine() قرار دهید. مانند دو نمونه ی زیر

```
class Exercise
{
  static void Main()
  {
    System.Console.WriteLine(248);
    System.Console.Write(1);
  }
}
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

اگر داخل پرانتز System.Console.WriteLine() را خالی بگذارید، خطی تهی نمایش داده می شود.

نمایش های عددی

برنامه ی کامپیوتر عبارتند از مجموعه ای دستور که برای رایانه انجام کار مشخص، زمان اجرای آن و نحوه ی پیاده سازی آن را معیین می کند. این برنامه نویس است که دستورها را می نویسد. همان طور که پیش تر نیز بیان شد، می توان دستورات مزبور را در برنامه ی text edit یا، البته بر مبنای قوانین استاندارد زبان C# ولی با زبان شناس مانند انگلیسی نوشت. برای مثال، می توان دستوری نوشت که از کامپیوتر درخواست می کند عددی را در حافظه ذخیره کند. تمام دستوراتی که می نویسید باید به کامپیوتر منتقل شود.

مستحضر هستید که اشخاص دستورهای مختلفی می نویسند. همچنین، C# تنها زبان برنامه نویسی نیست، و برنامه نویسان سرتاسر دنیا از زبان های مختلف برای دادن دستور به رایانه استفاده می کنند. کامپیوتر برای درک تمام این زبان ها و تفسیر تمام دستورهایی که از این برنامه ها ارسال می شود، از زبان مختص به خود استفاده می کند که تمام برنامه ها باید آن را بشناسند و از آن پیروی کنند. زبان های داده شده آن جور که باید از مسئله ی فوق پیروی نمی کنند. برای مثال، C# که از زبان انگلیسی استفاده می کند، دستورات خود را به برنامه ی میانجی / واسط ای به نام Assembler تحویل می دهد.assembler دستورات را به نسخه ی فشرده تر تبدیل می کند که با وجود استفاده از لغات جدید، آن هم به زبان انگلیسی می باشد. کامپیوتر دوباره نسخه ی فشرده را به ورژنی ساده تر تبدیل می کند و در اختیار رایانه قرار می دهد.

زبان ساده شده ای که کامپیوتر استفاده می کند، متشکل از ترکیبی از ۱ ها و ۰ ها می باشد، یعنی assembler باید دستورات c# را به ۰ و ۱ تبدیل کند تا کامپیوتر بتواند آن را تفسیر کرده و اجرا کند.

سیستم های عددی

با سیستم هایی که از ده رقم ۰، ۱، ۲، ۳، ۴، ۵، ۶، ۷، ۸ و ۹ استفاده می کنند آشنایی دارید. ترکیبی از این ده عدد می تواند هر رقمی که شما دوست دارید بسازد. به این خاطر که زبان کامپیوتر از ۱۰ رقم استفاده می کند به آن سیستم decimal یا دهگانی می گویند.

همان طور که پیش تر ذکر شد، زبان کامپیوتر از ترکیب دو مقدار ۰ و۱ استفاده می کند. به چنین سیستمی، سیستم دوتایی یا binary می گویند. 1، 10، 1001 یا 1010110111 نمونه هایی از این سیستم می باشند. هر مقداری که به کامپیوتر داده می شود باید ترکیبی از ۰ و ۱ ها تبدیل شود.

همان طور که تصور می کنید نمایش دادن یک رقم بزرگ کار بسیار دشواری است. یک روش برای این منظور، از ۱۰عدد سیستم دهگانی و ۶ حرف از الفبای انگلیسی استفاده می کند. A، B، C، D، E، و F یا a، b، c، d، e و f. به این معنا که سیستم فوق از ۱۶ کاراکتر برای ایجاد یک

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

رقم استفاده می کند. به همین دلیل است که به آن سیستم hexadecimal یا مبنای ۱۶می گویند. برای این که فرایند تشخیص یک رقم مبنای ۱۶ از کلمه آسان شود، رقم داده شده باید با 0x آغاز شود.نمونه های آن به این صورت است. x2۰، 0xED8، 0x962AAED3.

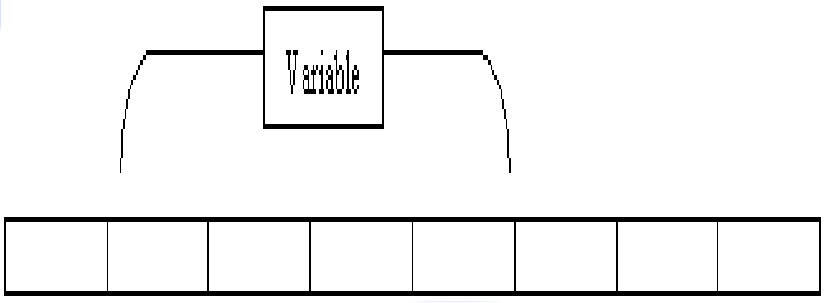
علامت دار و بدون علامت

یک از ویژه گی های رقم این است که مشخص می کند آیا از ۰ کوچکتر است، با آن برابر است یا از آن بزرگتر است. عددی که کوچکتر از صفر است منفی محسوب می شود. چنین رقمی به دنبال علامت – می آید. چنانچه، عددی از صفر بزرگتر بود مثبت تلقی می گردد. چنین رقمی به دنبال علامت + می آید. به ارقامی که یکی از این دو علامت را به یدک می کشند signed یا علامت دار می گویند. هر عددی هم که هیچ یک از این دو علامت را نداشته باشد unsigned یا مثبت اتلاق می گردد. برای مثال، عدد ۰ بدون علامت یا unsigned است.

تعریف متغییرها

همان طور که پیش تر ذکر شد، برای ذخیره سازی مقدار (value) معینی در حافظه ابتدا باید مقدار حافظه ی مورد نیاز را مشخص کنید، سپس باید برای آن اسم انتخاب کنید. مقدار حافظه ی مورد نیاز را data type (نوع داده) می گویند. مقدارهای متفاوت، مقدار حافظه ی متفاوت می طلبد. این را هم گفتیم که رایانه مقدار را ۰ و ۱ می بیند. آن قسمت حافظه که تنها یک ۰ یا ۱ در خود جای می دهد bit خوانده می شود.

تصور کنید که bit یک شئ تهی است که می تواند خالی یا پر باشد.



هنگامی که آن شی خالی است، bit مقدار ۰ را نشان می دهد و زمانی که پر است، مقدار ۱.

تعریف متغییر

با به کاربردن ترکیبی از bit ها، به compiler پیغام می دهید که به متغییر نیاز دارید. از چنین فرایندی به عنوان (declaring a variable) تعریف متغییر یاد می شود. همان طور که قبل توضیح داده شد، مقدار حافظه ای که به آن نیاز دارید data type گفته می شود. بنابراین،

هنگام تعریف متغییرباید نوع داده ی (data type) دلخواه را مشخص کنید. در C#، نوع داده (data type) با یک کلیدواژه نمایش داده می شود.

برای تعریف متغییر دو راه پیش رو دارید.

- اگر نوع مقداری که مایلید در حافظه ذخیره شود را می دانید، از فرمول زیر پیروی کنید.

DataTypeVariableName;

- اگر نمی خواهید نوع مقدار را مشخص کنید، گزینه های دیگری برای شما فراهم است.

  می توانید از کلید واژه ی var استفاده کنید.

  می توانید کلید واژه ی object را به کار ببرید.

  می توان از کلیدواژه ی dynamic نیز استفاده کرد.

مقدار دهی اولیه ی متغییر

همان طور که از اسم آن پیدا است، مقدار دهی اولیه عبارتند از ذخیره سازی مقدار اولیه در جای (مشخص آن.) بیش تعریف شده به منظور اختصاص دادن متغییر.

نوع داده، به دنبال آن اسم متغییر، علامت =، مقدار دلخواه ولی مناسب را تایپ کنید.

```
class Exercise
{
  static void Main()
  {
    DataTypeVariableName = DesiredValue;
  }
}
```

با این روش، شما می توانید متغییر را در ابتدای کار با مشخص کردن اسم و نوع داده تعریف کنید و در انتهای آن علامت نقطه ویرگول قرار دهید. سپس، در خطی دیگر، اسم متغییر را تایپ کرده و به دنبال آن علامت =، و مقدار مورد نظر را وارد کنید.

```
class Exercise
{
  static void Main()
  {
    DataTypeVariableName;
    VariableName = DesiredValue;
```

```
  }
}
```

ابتدا کلیدواژه ی var را تایپ کرده و به دنبال آن، اسم متغییر، علامت = و مقدار معین را درج کنید.

```
class Exercise
 {
   static void Main()
   {
     varVariableName = DesiredValue;
   }
 }
```

این بار باید تمام مراحل را یکجا انجام دهید. در این مورد، نمی توان ابتدا اسم متغییر را با قرار دادن var و نقطه ویرگول (;) تعریف کرده و بعد (در خطی دیگر) مقدار دهی اولیه متغییر را انجام دهید. با این کار، پیغام خطا دریافت می کنید.

```
class Exercise
{
  static void Main()
  {
    varVariableName;

    VariableName = DesiredValue; // Error
  }
}
```

دو گزینه ی پیش روی دیگر نیز، استفاده از کلیدواژه های object یا dynamic می باشد.

پس از تعریف و مقدار دهی اولیه ی متغییر، compiler مقدار آن متغییر را حافظه ای که برای آن اختصاص یافته ذخیره می شود. حال، می توان به آن مقدار دسترسی پیدا کرد و در صورت نیاز آن را تغییر داد.

مقدار تهی (null value)

می توان با اضافه کردن علامت سوال (؟) به نوع داده (data type) متغیر موردنظر، متغیر را طوری تعریف کنید که مقدار تهی (null value) داشته باشد. برای این منظور زبان C# کلیدواژه ی null را در اختیار شما قرار می دهد.

می توان هنگام تعریف متغیر، مقدار null را به آن اختصاص داد.

```
class Exercise
{
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
  public static void Main()
  {
    DataType?VariableName= null;
  // You can use the variable
  }
}
```

همچنین می توان این مقدار را پس از تعریف متغیر اختصاص داد.

```
class Exercise
{
  public static void Main()
  {
  DataType?VariableName;
  VariableName= null;
  // You can use the variable
  }
}
```

آنچه اهمیت دارد آن است که مقدار پیش از به کاربردن متغیر به آن اختصاص داده شود.

Byte

ترکیبی از چهار bit

اگرچه یک bit قابل دسترس و استفاده است، باید به خاطر داشت که نمی توان در آن مقدار ذخیره کرد به این معنا که نمی توان از compiler خواست که مقدار ۱ را در یک bit ذخیره کند. این امر به این خاطر است که، حتی کوچکترین مقدار C# نیز به بیش از یک bit برای ذخیره شدن نیاز دارد. ترکیبی کوچکتر از چهار bit وجود ندارد (در برخی زبان ها یا پیاده سازی زبان assembly به ترکیب چهار bit، nibble گفته می شود).

با ایجاد ترکیبات چهارتایی از پر (۱) و تهی (۰)، همگی ۱۶ ترکیب به دست می آید.

در این ترکیبات چهارتایی، bit ها ۰، ۱، ۲ و ۳ شمرده می شوند. bit ای که در راست ترین کناره یا موقعیت قرار دارد (۰)، bit رده پایین خوانده می شود (LOBIT).

آخرین bit یا bit ای که در چپ ترین موقعیت قرار دارد (۳)، bit رده بالا خوانده می شود (HIBIT).

اگر بخواهیم ترکیبات چهار bit ای را با سیستم دودویی (binary) نمایش دهیم به این نتایج دست می یابیم ۰۰۰۰، ۰۰۰۱، ۰۰۱۰، ۰۰۱۱، ۰۱۰۰، ۰۱۰۱، ۰۱۱۰، ۰۱۱۱، ۱۰۰۰، ۱۰۰۱، ۱۰۱۰، ۱۰۱۱، ۱۱۰۰، ۱۱۰۱، ۱۱۱۰، ۱۱۱۱، که همگی ۱۶ ترکیب به دست می آید. با فرمت دهدهی (decimal) ترکیبات بالا این نتایج را به دست می دهد : ۰، ۱، ۲، ۳، ٤، ٥، ٦، ٧، ٨، ٩، ١٠، ١١، ١٢، ١٣، ١٤، و ١٥. حال، با فرمت hexadecimal (مبنای ۱۶)

Decimal	Binary	Hexadecimal
0	0000	0x0
1	0001	0x1
2	0010	0x2
3	0011	0x3
4	0100	0x4
5	0101	0x5
6	0110	0x6
7	0111	0x7
8	1000	0x8
9	1001	0x9
10	1010	0xA
11	1011	0xB
12	1100	0xC
13	1101	0xD

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

14	1110	0xE
15	1111	0xF

جدول تبدیل عددی

حداقل و حداکثر مقدارها در ترکیب چهار bit ای به این صورت است.

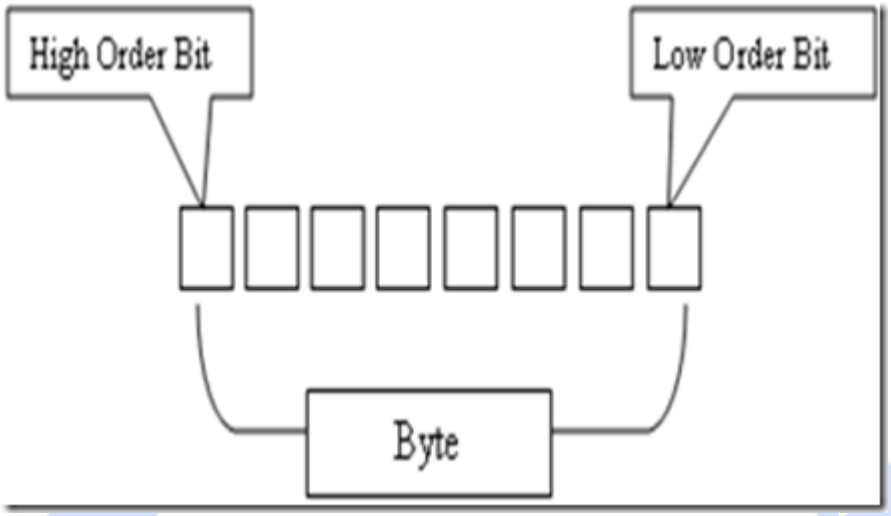
	Decimal	Hexadecimal	Binary
Minimum	0	0x0	0000
Maximum	15	0xf	1111

البته، باید در نظر داشته باشید که با ترکیبات چهارتایی هیچ کاری نمی توان کرد. زیرا ترکیبات بیان شده بسیار کوچک هستند و گنجایش ذخیره هیچ چیز را ندارد. به دو دلیل درباره ی آن بحث کردیم : اول اینکه، باید بدانید که عدد ۱۶ از کجا سر رشته می گیرد. دوم اینکه، پیش زمینه ی معرفی byte را فراهم آورد.

ترکیبی از ۸ بیت

یک byte ترکیبی از ۸ bit است که کنار هم قرار گرفته. اولین bit (bit ای که در راست ترین قسمت قرار گرفته) ۰ و آخرین bit (چپ ترین bit) ۷ می باشد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



۰ کم اهمیت ترین bit است که به آن LOBIT نیز می گویند. ۷ مهم ترین bit محسوب می شود و نام دیگر آن HIBIT است.

ترکیبات ۸ تایی با سیستم binary: ۰۰۰۰۰۰۰۰، ۰۰۰۰۰۰۰۱، ۰۰۰۰۰۰۱۰، ۰۰۰۰۰۰۱۱، ۰۰۰۰۰۱۰۰، ۰۰۰۰۰۱۰۱، ۰۰۰۰۰۱۱۰، ۰۰۰۰۰۱۱۱، ۰۰۰۰۱۰۰۰، ۰۰۰۰۱۰۰۱، ۰۰۰۰۱۰۱۰، ۰۰۰۰۱۰۱۱، ۰۰۰۰۱۱۰۰، ۰۰۰۰۱۱۰۱، ۰۰۰۰۱۱۱۰، ۰۰۰۰۱۱۱۱ تا ۱۱۱۱۱۱۱۱. نمایش دادن یک عدد بزرگ با فرمت دودویی، خواندن آن را سخت می کند. روش مناسب تر، دسته بندی آن ها در گروه های چهارتایی است به این صورت ۱۰۱۱ ۰۱۰۰ = ۰۱۰۰۱۰۱۱

به منظور ارزیابی تعداد ترکیبات در فرمت دهدهی (decimal)، عدد ۲ (که نشانگر decimal است)، را به توان bit موردنظر (۰، ۱، ۲، ۳، ٤، ٥، ٦، 7) برده، سپس عدد را اضافه می کنیم به این صورت

$$27 + 26 + 25 + 24 + 23 + 22 + 21 + 20$$

$$= 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1$$

$$= 255$$

بنابراین، ۲٥٥ ترکیب احتمالی ۸ bit ای داریم. ترکیبات بالا را می توان با فرمت hexadecimal نیز حساب کرد : 0x10، 0x2...، 0xA ، up to 0xFFFF، ...، 0xC8، ...، 0xA1

محاسبات در سه سیستم عددی مختلف

	Decimal	Hexadecimal	Binary

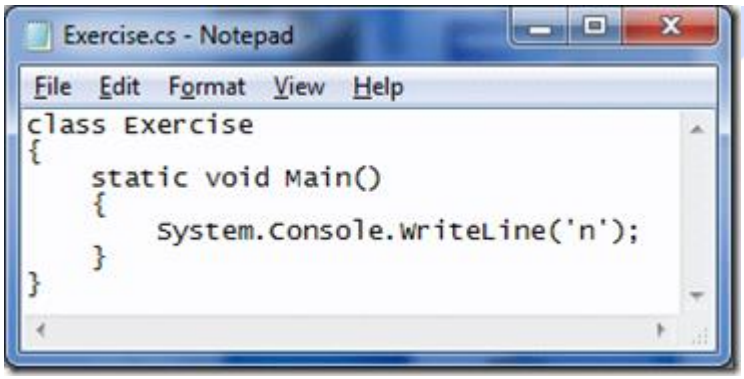
آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

Minimum	0	0x0	0000
Maximum	255	0xff	1111 1111

حداقل مقدار حافظه اختصاص داده شده (توسط intel computer) به منظور ذخیره سازی byte می باشد. همان طور که مستحضر هستید، یک byte متشکل از ۸ bit متوالی و کنارهم قرار گرفته است. مقدار حافظه ای که به اندازه ی یک byte در اختیار شما قرار داده می شود، تنها گنجایش ذخیره کردن یک نشانه (که بر روی صفحه کلید درج شده است) را دارد. این نشانه ها، که به آن ها کاراکتر هم گفته می شود، توسط ASCII، کد استاندارد آمریکایی به منظور تبادل اطلاعات تعبیه شده اند. اما باید در نظر داشت که ASCII از تنها ۱۲۸ عدد decimal (بر مبنای فرمت ۷ bit ای) استفاده می کند (از ۰ تا ۱۲۷).

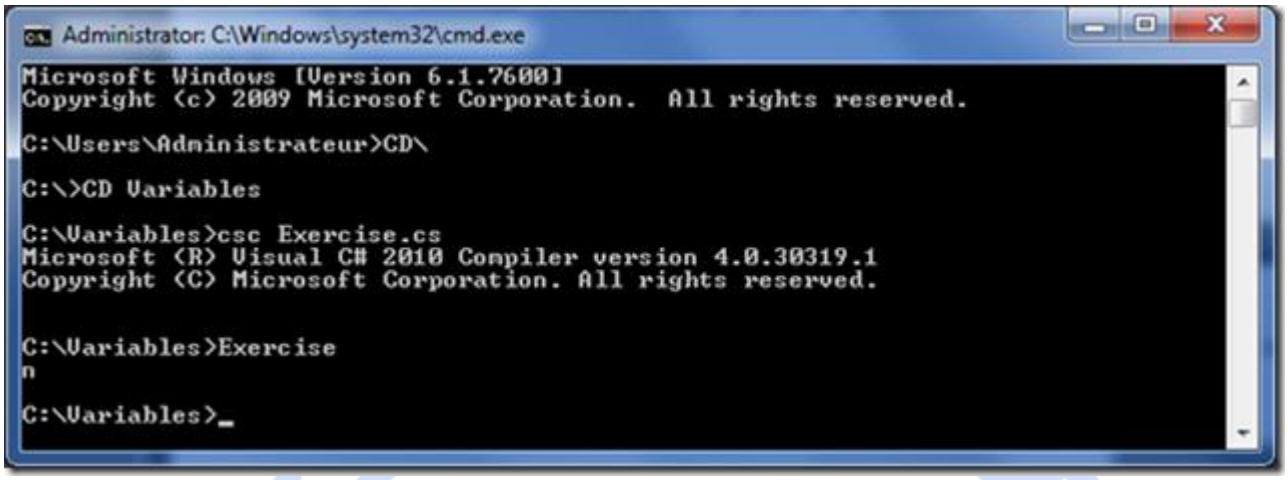
کلیه ی کاراکترهایی که در صفحه کلید مشاهده می کنید، به عنوان یک مقدار عددی نمایش داده می شود، اما هریک از این نشانه ها چه عدد، چه حرف و چه علامت همگی کاراکتر محسوب می شوند. برای نمایش دادن هر کاراکتری در صفحه ی نمایش، می توان آن را با Write() یا WriteLine()، ارسال کرد (pass) و کاراکتر مورد نظر را در علامت (' ') قرار داد.

```
class Exercise
{
  public static void Main()
  {
    System.Console.WriteLine('n');
  }
}
```



فایل موردنظر با نام Exercise.cs در فولدر Variables در درایو C:\ ذخیره گشت. پس از ترجمه (compile) و اجرا شدن، حرف n در صفحه نمایش داده می شود.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



کاراکترها

در الفبای زبان انگلیسی، حرف به یکی از نشانه های زیر گفته می شود : a، b، c، d، e، f، g، h، i، j، k، l، m، n، o، p، q، r، s، t، u، v، w، x، y ، z، A، B، C، D، E، F، G، H، I، J، K، L، M، N، O، P، Q، R، S، T، U، V، W، X، Y و Z. جدا از این کاراکترهای خوانا، به علامت های ذیل نیز رقم (digit) می گویند : ۰، ۱، ۲، ۳، ٤، ٥، ٦، ۷، ۸، ۹. علامت های دیگری وجود دارند که به آن ها نیزکاراکتر می گویند : ` ~ ! @ # $ % ^ & * ( ) - _ = + [{] } \ | ; : ' > < , / . ? ".

زبان C# هر چیزی را که بتوان به صورت نشانه ای به نمایش گذاشت را یک کاراکتر به حساب می آورد. به منظور تعریف متغیری که مقدارش یک کاراکتر می باشد، می توان از کلیدواژه ی var استفاده کرد و متغیر را با کاراکتری درون علامت ' ' مقدار دهی اولیه (initialize) کرد. به مثال زیر توجه کنید.

```
class Exercise
{
  public static void Main()
  {
     var gender = 'F';
     System.Console.Write("Student Gender: ");
     System.Console.WriteLine(gender);
  }
}
```

نتیجه ی زیر حاصل می گردد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

همچنین، می توان از کلیدواژه ی char استفاده کرد. توجه خود را به مثال زیر جلب کنید.

```
class Exercise
{
  public static void Main()
  {
    char gender = 'M';
    System.Console.Write("Student Gender: ");
    System.Console.WriteLine(gender);
    System.Console.ReadKey();
  }
}
```

نکته

تمام زبان هایی که بر مبنای C فعالیت می کنند نوع داده ی char را پشتیبانی می کنند.

نتیجه ی زیر به دست می آید.

```
Student Gender: M
```

Escape Sequence

Escape sequence کاراکتری ویژه است که در صفحه نمایش قابل رویت نمی باشد. برای نمونه می توان از این کاراکتر برای شروع نوشتن دستور در خط بعدی استفاده کرد (به برنامه فهماند که این خط به اتمام رسیده و خط دیگری باید شروع شود). کاراکتر مزبور با این علامت /

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

نشان داده می شود که به دنبال آن ممکن است یک کاراکتر یا نشانه ی دیگر قرار گیرد. برای مثال، escape sequence ای که در خط بعدی قرار می گیرد به این صورت نمایش داده می شود : \n

کاراکتر escape sequence ممکن است داخل ' ' قرار داده شود : '\n'. هم چنین این کاراکتر را می توان داخل " " قرار داد : "\n".

نکته

تمام زبان هایی که مبنای آن C هست، این کاراکتر ها را پشتیبانی می کنند.

Escape Sequence	Name	Description
\a	Bell (alert)	صدایی از کامپیوتر پخش می کند
\b	Backspace	نشان گر موس را به عقب می برد
\t	Horizontal Tab	نشان گر موس را به صورت افقی به حرکت در می آورد
\n	New line	نشان گر موس را به آغاز خط بعدی می برد
\v	Vertical Tab	جدول بندی عمودی ایجاد می کند
\f	Form feed	
\r	Carriage return	فرایند بازگشت به ابتدای خط را فراهم می سازد
\"	Double Quote	(") علامت نقل و قول را نمایش می دهد
\'	Apostrophe	(') علامت آپستروف را نمایش می دهد
\?	Question mark	علامت سوال را نشان می دهد
\\	Backslash	را نشان می دهد (\) علامت
`	Null	کاراکتر تهی را نمایش می دهد

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

برای استفاده از Escape sequence، همچنین می توان یک متغیر char معرفی کرده، آن را با کاراکتر escape sequence دلخواه مقداردهی اولیه کرد، سپس داخل '' قرار داد.

نوع داده ی Byte

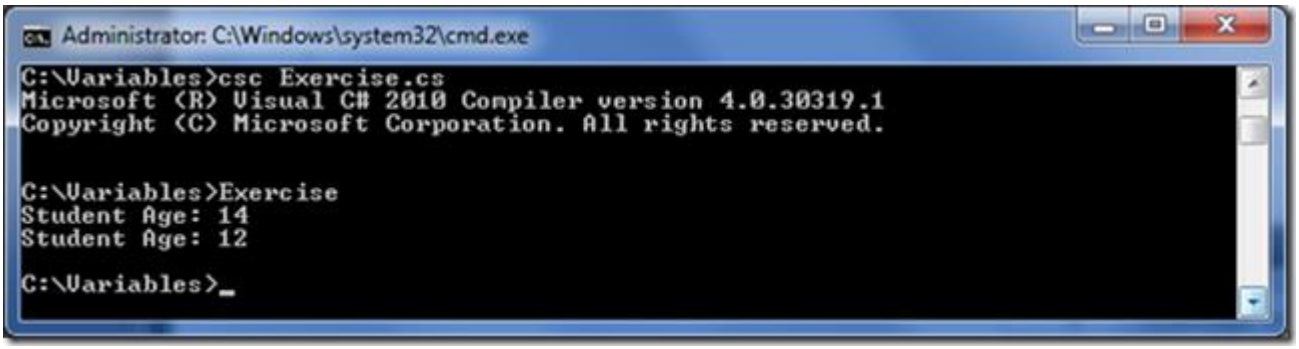
یک byte عددی بی علامت (unsigned number) است که مقدار آن از ۰ تا ۲۵۵ متغیر می باشد، به این خاطر هیچ چیز در یک byte قابلیت ذخیره شدن ندارد. برای معرفی متغیری که دربردارنده ی یک عدد طبیعی کوچک است، می توان از کلیدواژه ی byte استفاده کرد مانند مثال زیر

byte Age

می توان متغیر byte را در حین معرفی متغیر یا پس از انجام فرایند (معرفی) مقدار دهی اولیه کرد. مثال زیر نوع داده ی byte را به کاربرده.

```
class Exercise
{
  public static void Main()
  {
    byte age = 14;
    System.Console.Write("Student Age: ");
    System.Console.WriteLine(age);
    age = 12;
    System.Console.Write("Student Age: ");
    System.Console.WriteLine(age);
  }
}
```

که محصول زیر را به دست می دهد.



به هیچ وجه از مقداری بالای ۲۵۵ برای متغیر byte استفاده نکنید، زیرا در آن صورت error دریافت می کنید. همچنین، می توان از کلیدواژه ی var برای معرفی متغیر استفاده کرد، سپس آن را با عددی کوچک مقداردهی اولیه کرد. به این مثال توجه کنید.

```
class Exercise
{
  public static void Main()
  {
      var age = 14;
      System.Console.Write("Student Age: ");
      System.Console.WriteLine(age);
      age = 12;
      System.Console.Write("Student Age: ");
      System.Console.WriteLine(age);
     System.Console.ReadKey();
  }
}
```

به جای عدد دهدهی (decimal number)، می توان متغیر انتگرال را با مقدار hexadecimal (مبنای۱۶) مقداردهی اولیه کرد. ابتدا، اطمینان کسب کنید که معادل دهدهی (decimal equivalent) از مقدار ۲۵۵ پایین تر است. به مثال زیر توجه کنید.

```
class Exercise
  {
    public static void Main()
    {
        var number = 0xFE;
        System.Console.Write("Number: ");
        System.Console.WriteLine(number);
       System.Console.ReadKey();
    }
  }
```

نتیجه ی زیر حاصل می گردد.

```
Number: 254
Press any key to continue...
```

استفاده کردن از Byte

فایل Program.cs را به این صورت تغییر دهید.

```
class Order
{
  static void Main()
  {
    byte? shirts = null;
    byte? pants = null;
```

```
        shirts = 4;
        pants = 1;
        System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
        System.Console.WriteLine("==========================");
        System.Console.WriteLine("Item Type  Qty");
        System.Console.WriteLine("-------------------------");
        System.Console.Write("Shirts    ");
        System.Console.WriteLine(shirts);
        System.Console.Write("Pants     ");
        System.Console.WriteLine(pants);
        System.Console.WriteLine("==========================");
        System.Console.ReadKey();
    }
}
```

به منظور اجرای برنامه به main menu مراجعه کرده، سپس گزینه ی Debug -> Start Debugging را انتخاب کنید. محصول زیر به دست می آید.

```
-/- Georgetown Cleaning Services -/-
==========================
Item Type  Qty
-------------------------
Shirts    4
Pants     1
==========================
```

حال، برای بستن پنجره ی DOS دکمه ی Enter را فشار دهید.

Byte علامت دار

یک عدد byte، زمانی علامت دار معرفی می شود که بتواند مقداری مثبت یا منفی از ۱۲۸- تا ۱۲۷ را در خود نگه دارد، این مقدار در یک byte ذخیره می شود. به منظور معرفی متغیری برای مقدار گفته شده، از کلیدواژه ی sbyte استفاده کنید. نمونه ی آن را در زیر مشاهده می کنید.

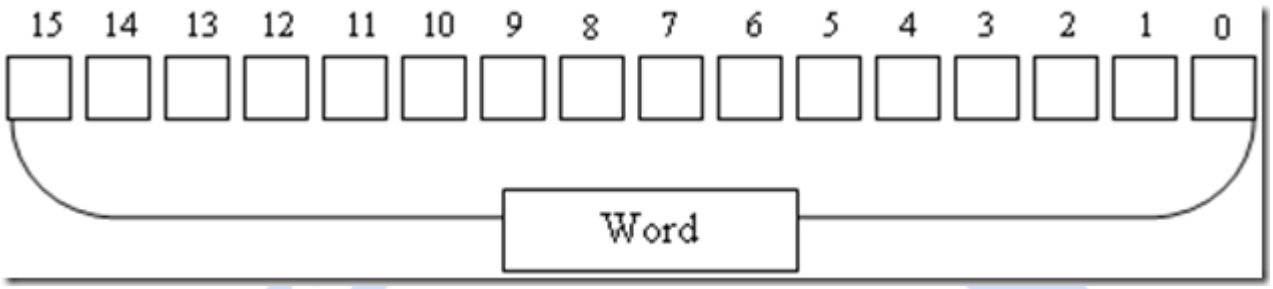
```
class Exercise
{
    static void Main()
    {
        sbyte roomTemperature = -88;
        System.Console.Write("When we entered, the room temperature was ");
        System.Console.WriteLine(roomTemperature);
    }
}
```

که نتیجه ی زیر از آن به دست می آید.

```
the room temperature was -88،When we entered
```

واژه (Word)

یک واژه متشکل از ۱۶ bit متوالی است. bit ها از راست به چپ شمرده می شوند (از ۰).



راست ترین bit یک واژه ۰ می باشد که به آن کم اهمیت ترین bit یا bit سطح پایین و یا LOBIT می گویند. چپ ترین bit، ۱۵، مهم ترین bit یا bit سطح بالا و یا HIBIT خوانده می شود. bit های دیگر بسته به موقعیتی که در آن قرار گرفته اند تعریف می شوند : ۱، ۲، ۳ و غیره.

نظر به این که یک واژه از دو byte تشکیل شده، گروه 8 bit سمت راست همان LOBYTE خوانده می شود و دیگر گروه 8 bit ای که در سمت چپ قرار می گیرد HIBYTE نام دارد.

نمایش یک واژه با فرمت (قالب) دودویی (binary) به این صورت است : 0000000000000000. برای خواناتر کردن آن، می توان bit ها را در گروه های چهارتایی قرار داد به این صورت : 0000 0000 0000 0000. بنابراین، کمترین مقدار دودویی که واژه قادر به نمایش دادن آن است به این شکل خواهد بود : 0000 0000 0000 0000. کمترین مقدار دهدهی (decimal) یک واژه معادل ۰ می باشد. کمترین مقدار مبنای ۱۶ (hexadecimal) که می توان در یک واژه ذخیره کرد برابر با 0x0000000000000000 می باشد که البته به این اشکال نیز نمایش داده می شود : 0x00000000 یا 0x0000 و یا 0x0. تمام این ارقام یک مقدار را به دست می دهند و آن 0x0 است.

حداکثر مقدار دودویی (binary) که یک واژه می تواند نشان دهد معادل : 1111 1111 1111 1111 می باشد. برای به دست آوردن حداکثر مقدار دهدهی (decimal) یک واژه ، می توانید از فرمول پایه ۲ استفاده کنید و به جای هر bit عدد ۱ را قرار دهید.

1*215+1*214+1*213 + 1*212 + 1*211 + 1*210 + 1*29 + 1*28 + 1*27 + 1*26 + 1*25 + 1*24 + 1*23 + 1*22 + 1*21 + 1*20

= 32768 + 16384 + 8192 + 4096 + 2048 + 1024 + 512 + 256 + 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1

= 65535

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

برای به دست آوردن حداکثر مقدار یا رقم مبنای ۱۶ (hexadecimal) که در یک واژه قابلیت ذخیره شدن را داشته باشد، تمام گروه های 4 bit ای را با f یا F جایگزین کنید.

1111	1111	1111	1111
f	f	f	f

= 0xffff

= 0xFFFF

= 0Xffff

= 0XFFFF

short integers

یک واژه (word)، که متشکل از 16 bit همجوار یا 2 byte می باشد، می تواند یک عدد طبیعی (natural number) را در خود جای دهد. همان طور که قبلاً ذکر شد، بیشترین مقدار عددی (numeric value) که در یک واژه می توان ذخیره کرد 65535 است. به منظور تعریف متغیری برای این مقدار، می توان کلیدواژه ی var را به کاربرد و متغیر نام برده را با مقداری از -32768 تا 32767 مقداردهی اولیه کرد. به مثال زیر توجه کنید.

```
class Exercise
{
  static void Main()
  {
    var schoolEffective = 1400; // Number of Students
    System.Console.Write("School Effective: ");
    System.Console.WriteLine(schoolEffective);
  }
}
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

نتیجه ی زیر حاصل می گردد.

```
School Effective: 1400
Press any key to continue...
```

(به این خاطر که byte تنها گنجایش کاراکترها و ارقام کوچک را دارد)، هر زمان قصد داشتید عددی را در برنامه ی خود به کار ببرید، توجه داشته باشید که کوچکترین نمودی که می توان به کاربرد واژه است.

عدد طبیعی (natural number) را integer نیز می گویند. کوچکترین (integer) را تنها با کمک کلیدواژه ی short می توان در یک واژه ذخیره کرد. به دلیل این که short integer به صورت پیش فرض علامت دار می باشد، می تواند مقداری که از -32768 تا 32767 متغیر است را در خود ذخیره کند. در زیر نمونه ی برنامه ای که دو short integer را به کار می برد مشاهده می کنید.

```
class Exercise
{
  static void Main()
  {
    short numberOfPages;
    short temperature;
    numberOfPages = 842;
    temperature = -1544;
    System.Console.Write("Number of Pages of the book: ");
    System.Console.WriteLine(numberOfPages);
    System.Console.Write("Temperature to reach during the experiment: ");
    System.Console.Write(temperature);
    System.Console.WriteLine(" degrees\n");
  }
                                        }
```

نتیجه ی زیر به دست می آید.

```
Number of Pages of the book: 842
Temperature to reach during the experiment: -1544 degrees
```

(به دلیل این که short integer ها می توانند ارقام (number) بزرگتر از byte علامت دار را در خود جای دهند)، هر مقداری که برای byte علامت دار تعریف می کنید را می توان برای متغیر short هم تعریف کرد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

Short integer های بدون علامت

از متغیری که دربردارنده ی اعداد مثبت و نسبتاً کوچکی است با نام unsigned short integer یاد می‌شود. چنین متغیری را می توان با کلیدواژه های ushort یا var تعریف کرد. short integer ها ی بدون علامت می توانند اعدادی که در برد ۰ تا 65535 قرار دارند را در برگیرد (به همین دلیل است که در تنها ۱۶ bit جای می گیرند). مثال های آن را در زیر مشاهده می کنید

```
class Exercise
{
  static void Main()
  {
    // These variables must hold only positive integers
    ushort numberOfTracks;
    ushort musicCategory;
    numberOfTracks = 16;
    musicCategory = 2;
    System.Console.Write("This music album contains ");
    System.Console.Write(numberOfTracks);
    System.Console.WriteLine(" tracks");
    System.Console.Write("Music Category: ");
    System.Console.Write(musicCategory);
    System.Console.WriteLine();
  }
}
```

نتیجه ی زیر به دست می آید.

```
This music album contains 16 tracks
Music Category: 2
```

به کاربردن short integer های بدون علامت

برای به کاربردن short integer ها، فایل را به صورت زیر تغییر دهید.

```
class Order
{
  static void Main()
  {
    byte? shirts = null;
    byte? pants = null;
    ushort? otherItems = null;
    shirts = 4;
    pants = 1;
    otherItems = 3;
```

```
        System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
        System.Console.WriteLine("==========================");
        System.Console.WriteLine("Item Type  Qty");
        System.Console.WriteLine("-------------------------");
        System.Console.Write("Shirts     ");
        System.Console.WriteLine(shirts);
        System.Console.Write("Pants      ");
        System.Console.WriteLine(pants);
        System.Console.Write("Other Items ");
        System.Console.WriteLine(otherItems);
        System.Console.WriteLine("==========================");
        System.Console.ReadKey();
    }
}
```

به منظور اجرای برنامه، به main menu مراجعه کرده و روی Debug -> Start Debugging کلیک کنید. نتیجه ی زیر حاصل می گردد.

```
-/- Georgetown Cleaning Services -/-
==========================
Order Date: 7/15/2002
-------------------------
Item Type  Qty
-------------------------
Shirts     4
Pants      1
Other Items 3
==========================
```

کلید Enter را بزنید تا پنجره ی DOS بسته شود.

حال، محیط برنامه نویسی را نیز ببندید.

هنگامی که از شما پرسیده شد، می خواهید SAVE کنید یا نه، NO را بزنید.

مواجهه با مقادیر بسیار بزرگ

Double-word (گروه دوکلمه) عبارتند از گروهی متشکل از دو واژه ی متوالی. به عبارت دیگر، گروه دوکلمه ترکیبی از 4 byte یا 32 bit می باشد. bit ها که از راست به چپ شمرده می شوند، از ۰ آغاز شده و به ۳۱ ختم می شوند. راست ترین bit، bit 0، کم اهمیت ترین bit یا LOBIT تلقی می گردد و چپ ترین bit، bit 31، مهم ترین bit یا HIBIT خوانده می شود. بقیه ی bit ها، بسته به موقعیتی که در آن قرار گرفته اند تعریف می شوند.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

گروه 8 bit اول (از ۰ تا ۷)، که byte سمت راست حساب می شود، LOBYTE خوانده می شود. گروه 8 bit آخر (از ۲۴ تا ۳۱) که byte سمت چپ محسوب می شود، HIBYTE اطلاق می گردد. byte های دیگر نیز بسته موقعیتی که در آن قرار گرفته اند، تعریف می شوند. حال، گروه 16 bit ای سمت راست یا همان واژه ی سمت راست، LOWORD خوانده می شود. گروه 16 bit ای سمت چپ، واژه ی سمت چپ، HIWORD گفته می شود.

حداقل عدد دودوئی (minimum binary number) که می توان با یک گروه دوکلمه ای (double-word) نشان داد ۰ می باشد. حداقل مقدار اعشاری (minimum decimal value) یک گروه دوکلمه معادل ۰ می باشد. برای بدست آوردن بیشینه ی مقدار اعشاری (maximum decimal value) یک واژه، می توان از فرمول پایه ی ۲ (base 2 formula) استفاده کرد و به هر bit، مقدار (value) ۱ داد.

2n-1	230	229	228	227	226	225	224
etc	74،073،1 824،1	9،870،536 12	4،435،268 56	7،217،134 28	8،108،67 64	4،554،33 32	2،777،16 16

223	222	221	220	219	218	217	216
608،388،8	304،194،4	152،097،2	576،048،1	288،524	144،262	072،131	536،65

215	214	213	212	211	210	29	28
768،32	384،16	192،8	096،4	048،2	024،1	512	256

27	26	25	24	23	22	21	20
128	64	32	16	8	4	2	1

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

1*231+1*230+1*229 + 1*228 + 1*227 + 1*226 + 1*225 + 1*224 + 1*223 + 1*222 + 1*221 + 1*220 + 1*219 + 1*218 + 1*217 + 1*216 + 1*215 + 1*214 + 1*213 + 1*212 + 1*211 + 1*210 + 1*29 + 1*28 + 1*27 + 1*26 + 1*25 + 1*24 + 1*23 + 1*22 + 1*21 + 1*20

432 + ،554،864 + 33،108،728 + 67،217،456 + 134،435،912 + 268،870،824 + 536،741،073،648 + 1،483،147،= 2

768 ،536 + 32،072 + 65،144 + 131،288 + 262،576 + 524،048،152 + 1،097،304 + 2،194،608 + 4،388،216 + 8،777،16

024 + 512 + 256 + 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1،048 + 1،096 + 2،192 + 4،384 + 8،+ 16

708،578،286،= 4

کم ترین مقدار مبنای ۱۶ (hexadecimal یا شانزده شانزدهی) که می توان در یک گروه دوکلمه ای جای داد (ذخیره کرد) 0x00000000000000000000000000000000 می باشد که در آخر برابر است با 0x0. به منظور به دست آوردن حداکثر (بیشینه) عدد مبنای ۱۶ که می توان با یک واژه نشان داد، باید هر گروه 4 bit ای را با (یک) f یا F جایگزین کرد.

1111	1111	1111	1111	1111	1111	1111	1111
f	f	f	f	f	f	f	f
= 0xffffffff = 0xFFFFFFFF							

به منظور تعریف متغییری که قادر است مقادیر بزرگ را در خود جای دهد، می توان از کلیدواژه ی var استفاده کرده و متغییر را با value دلخواه مقداردهی اولیه کرد.

```
class Exercise
{
  static void Main()
  {
    var population = 72394475;
    System.Console.Write("Country Population: ");
    System.Console.WriteLine(population);
  }
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

}

نتیجه ی زیر به دست می آید.

```
Country Population: 72394475
Press any key to continue...
```

به کار بردن integer های بدون علامت

ابتدا، برنامه ی Microsoft Visual Studio را اجرا کنید.

برای ایجاد برنامه ی کاربردی جدید (application)، به main menu مراجعه کرده و روی گزینه ی File -> New Project... کلیک کنید.

گزینه ی Empty Project را از لیست میانی انتخاب کنید.

اسم مورد نظر را به gdcs3 تغییر داده، سپس ok را کلیک کنید.

به منظور ایجاد فایل برای کد مورد نظر، فهرست گزینه ی اصلی (main menu) را باز کرده و روی Project -> Add New Item... کلیک کنید.

در لیست میانی گزینه ای است به نام Code File، آن را کلیک کنید.

اسم را به CleaningOrder تغییر دهید.

گزینه ی Add را انتخاب کنید.

در داکیومنت خالی که در اختیار شما قرار می گیرد، دستورات زیر را تایپ کنید.

```
class Order
 {
   static void Main()
   {
     byte? shirts = null;
     byte? pants = null;
     ushort? otherItems = null;
     shirts = 4;
     pants = 0;
     otherItems = 3;
     System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
     System.Console.WriteLine("============================");
     System.Console.WriteLine("------------------------");
```

```
    System.Console.WriteLine("Item Type  Qty");
    System.Console.WriteLine("-----------------------");
    System.Console.Write("Shirts    ");
    System.Console.WriteLine(shirts);
    System.Console.Write("Pants     ");
    System.Console.WriteLine(pants);
    System.Console.Write("Other Items ");
    System.Console.WriteLine(otherItems);
    System.Console.WriteLine("=========================");
    System.Console.ReadKey();
  }
}
```

۱۰. برنامه را اجرا کرده تا نتایج آن را مشاهده کنید.

Signed integers

گنجایش ذخیره ی اطلاعات یک گروه دوکمله ای، معادل با دو برابر حجم داده ای است که می توان در یک واژه ذخیره کرد. به عبارت دیگر، ظرفیت ذخیره سازی آن برابر است با 32 bit یا 4 byte و یا 4،294،967،295. بنابراین، از گروه دوکلمه ای برای ذخیره ی اعداد بزرگی استفاده می شود که در یک واژه جای نمی گیرد.

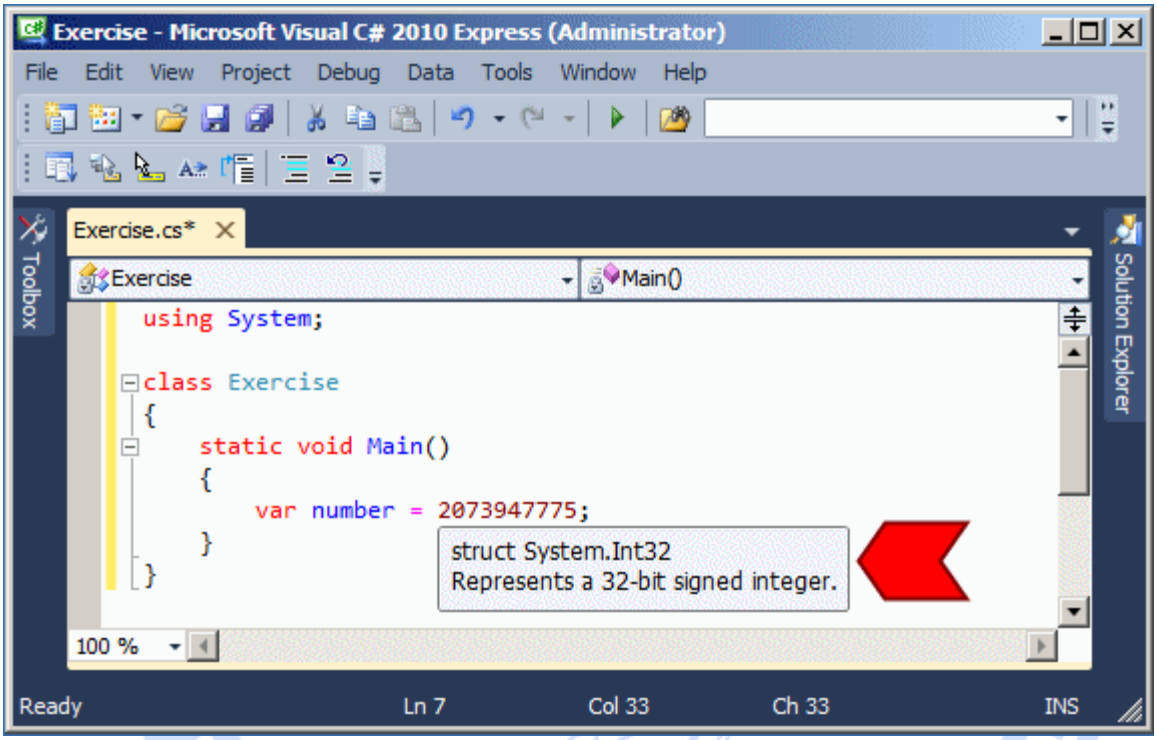
برای به کار بردن متغیری که قادر است اعداد بسیار بزرگ را در خود جای دهد، جدا از کلیدواژه ی var، می توان آن را با کلیدواژه ی int معرفی کرد. متغیری که به عنوان int معرفی می گردد، قادر است مقادیری که از 2،147،483،648 تا 2،147،484، 647 متغیر است، مثبت یا منفی، را در خود ذخیره کند (که همگی در 32 bit جای می گیرد). به مثال های زیر توجه کنید.

```
class Exercise
{
  static void Main()
  {
    int coordX;
    int coordY;
    coordX = 12;
    coordY = -8;
    System.Console.Write("Cartesian Coordinate System: ");
    System.Console.Write("P(");
    System.Console.Write(coordX);
    System.Console.Write(", ");
    System.Console.Write(coordY);
    System.Console.WriteLine(")\n");
  }
}
```

پس از اجرا، نتیجه ی زیر به دست می آید.

Cartesian Coordinate System: P(12، -8)

زمانی که متغیر integer ای را با کلیدواژه ی var تعریف کرده، سپس آن را با مقداری کم تر از 2،147،484،647 مقداردهی اولیه می کنید، compiler خودکار تصمیم می گیرد که حافظه ی مورد نیاز برای ذخیره ی متغیر مزبور 32 bit می باشد.



هنگام مقداردهی اولیه ی متغیر انتگرال (integral variable)، به جای عدد اعشاری (decimal number)، می توان از مقدار مبنای ۱۶ (hexadecimal value) استفاده کرد (که معادل دهدهی (decimal) آن کم تر از 2،147،484،647 است). نمونه ی آن را در زیر مشاهده می کنید.

```
class Exercise
{
  static void Main()
  {
    var number = 0xF0488EA;
    System.Console.Write("Number: ");
    System.Console.WriteLine(number);
  }
}
```

پس از اجرا، نتیجه ی زیر به دست می آید.

```
Number: 251955434
Press any key to continue...
```

Integer های بدون علامت

چنانچه متغیری تنها دربردارنده ی اعداد طبیعی مثبت باشد، می توان برای تعریف آن از کلیدواژه ی uint استفاده کرد (این کلیدواژه نشانگر intger بدون علامت می باشد). از کلیدواژه ی uint به منظور شناسایی integer مثبت 32 bit ای استفاده می شود که مقدار آن از 0 تا 4،294،967،295 متغیر است. نمونه های آن را در زیر مشاهده می کنید .

```
class Exercise
{
  static void Main()
  {
    uint dayOfBirth;
    uint monthOfBirth;
    uint yearOfBirth;
    dayOfBirth = 8;
    monthOfBirth = 11;
    yearOfBirth = 1996;
    System.Console.WriteLine("Red Oak High School");
    System.Console.Write("Student Date of Birth: ");
    System.Console.Write(monthOfBirth);
    System.Console.Write("/");
    System.Console.Write(dayOfBirth);
    System.Console.Write("/");
    System.Console.Write(yearOfBirth);
    System.Console.ReadKey();
  }
}
```

نتیجه ی زیر حاصل می گردد.

```
Red Oak High School
Student Date of Birth: 11/8/1996
```

به کاربردن integer های بدون علامت

به منظور استفاده از متغیرهای بدون علامت، فایل را به این ترتیب تغییر دهید.

```
class Order
{
  static void Main()
  {
    byte? shirts = null;
    byte? pants = null;
```

```
        ushort? otherItems = null;
        uint? orderDay = null;
        uint? orderMonth = null;
        uint? orderYear = null;
        shirts = 4;
        pants = 0;
        otherItems = 3;
        orderDay = 15;
        orderMonth = 7;
        orderYear = 2002;
        System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
        System.Console.WriteLine("=========================");
        System.Console.Write("Order Date: ");
        System.Console.Write(orderMonth);
        System.Console.Write('/');
        System.Console.Write(orderDay);
        System.Console.Write('/');
        System.Console.WriteLine(orderYear);
        System.Console.WriteLine("------------------------");
        System.Console.WriteLine("Item Type  Qty");
        System.Console.WriteLine("------------------------");
        System.Console.Write("Shirts     ");
        System.Console.WriteLine(shirts);
        System.Console.Write("Pants      ");
        System.Console.WriteLine(pants);
        System.Console.Write("Other Items ");
        System.Console.WriteLine(otherItems);
        System.Console.WriteLine("=========================");
        System.Console.ReadKey();
    }
}
```

۲. برنامه را اجرا کنید. نتیجه ی زیر را به دست می دهد.

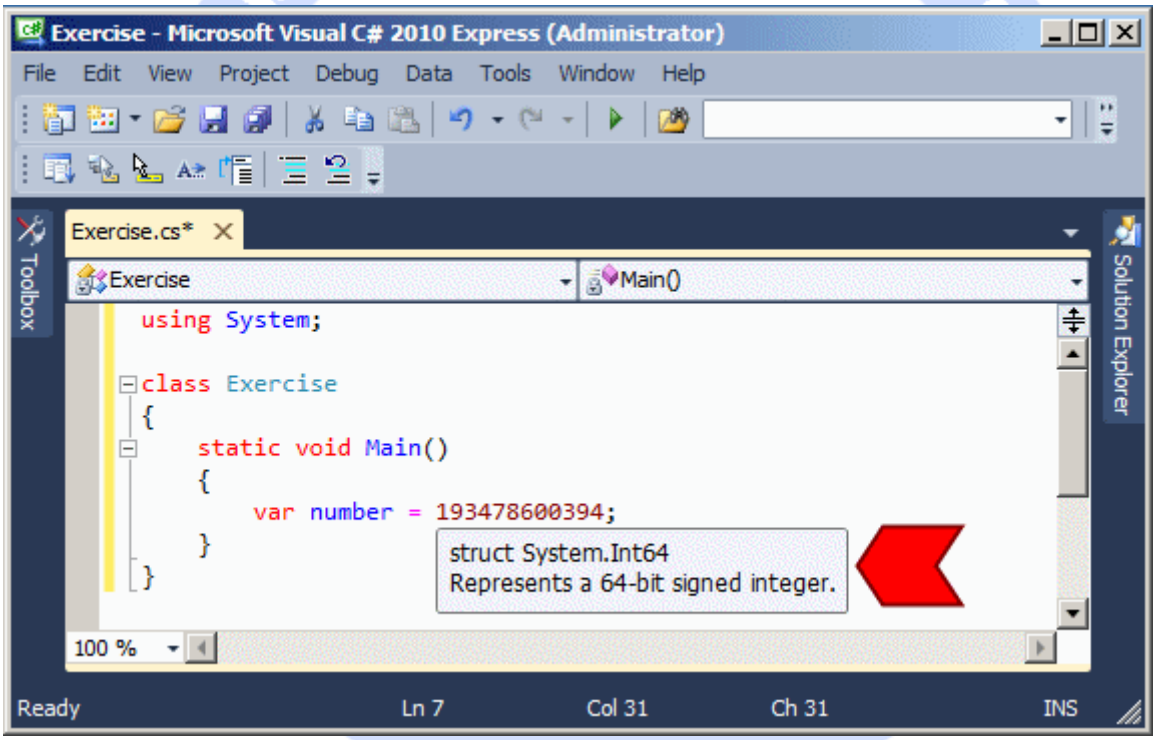
```
-/- Georgetown Cleaning Services -/-
    =========================
    Order Date: 7/15/2002
    ------------------------
    Item Type  Qty
    ------------------------
    Shirts     4
    Pants      0
    Other Items 3
    =========================
```

۳. پنجره ی DOS را ببندید.

چهارکلمه ای (Quad – word)

برای ذخیره سازی ارقام بزرگی که در گروه دو کلمه ای جا نمی شوند، می توان از ترکیب 64 bit ای استفاده کرد. از این ترکیب با عنوان گروه چهار کلمه ای (quad-core)، نیز یاد می شود. گروه چهار کلمه ای آن قدر بزرگ است که ارقامی که در محدوده ی 9-، 223، 372، 036، 854، 775، 808 تا 9، 223، 372، 036، 854، 775، 807 قرار می گیرد را می تواند در خود ذخیره کند.

چنانچه، متغیر integer ای را به وسیله ی کلیدواژه ی var تعریف کنید و آن را با value ای که از 2، 147، 484، 647 تا 9، 223، 372،036، 854، 775، 807 متغیر است مقداردهی اولیه کنید، compiler آن را در مقدار حافظه ی ٦٤ bit ذخیره می کند.



Long integers

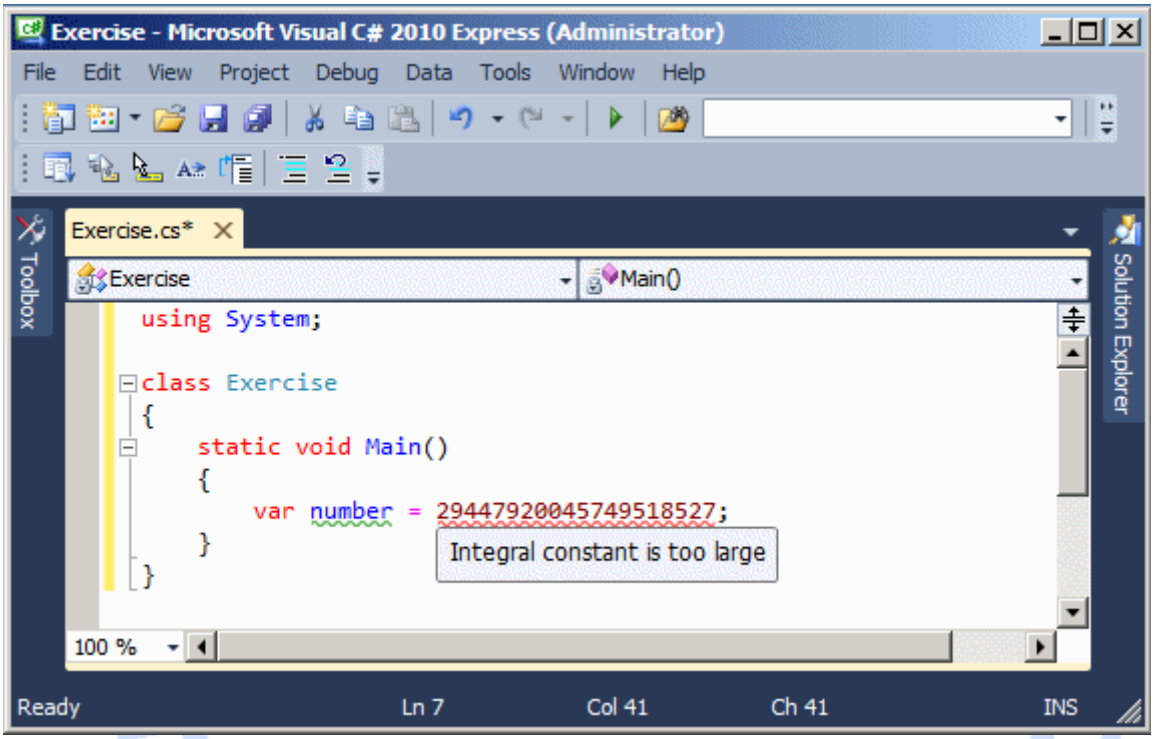
برای تعریف متغیری که گنجایش نگه داری ارقام بسیار بزرگ را داشته باشد (و تا 64 bit حافظه نیاز داشته باشد)، باید از کلید واژه های long و var استفاده کرد.

همان طور که پیش تر نیز گفته شد، چنانچه متغیری را با value ای کم تر از 2، 147، 484، 647 مقداردهی اولیه کنید، compiler تنها 32 bit حافظه به آن اختصاص می دهد. حال، چنانچه variable موردنظر را با value ای متغیر از 2، 147، 484، 647 تا 9، 223، 372، 036،

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

854، 775، 807 مقدار دهی اولیه کنید، compiler برای آن 64 bit حافظه کنار می گذارد. چنانچه مقدار مورد نظر از 9، 223، 372، 036،

854، 775، 807 بزرگ تر بود، compiler پیغام خطا صادر می کند.



به عبارت دیگر، باید مقدارهایی که به متغیر های انتگرال (integral variable) اختصاص می دهید را به 64 bit محدود کنید. به مثال زیر توجه کنید.

```
class Exercise
{
  static void Main()
  {
    var countryArea = 5638648;
    System.Console.Write("Country Area: ");
    System.Console.Write(countryArea);
    System.Console.Write("km2\n");
    System.Console.ReadKey();
  }
}
```

نتیجه ی زیر به دست می آید.

```
Country Area: 5638648km2
Press any key to continue...
```

می توان متغیر long را با value مبنای ۱۶ (hexadecimal) مقدار دهی اولیه کرد.

نوع داده ی long، همچنین میزان حافظه یا فضای دردسترس را نشان می دهد، ولی این امر بدین معنا نیست که باید حتماً از تمام فضا استفاده کرد. برای مثال، می توان از کلیدواژه ی long برای تعریف متغیری استفاده کرد در رینج عددی یکسان با نوع داده های short، int یا uint قرار دارد. پس اگر متغیری را به عنوان long معرفی کرده ولی از آن برای اعداد کوچکی که به 64 bit نیاز ندارند استفاده کنید، compiler خود میزان حافظه ی لازم برای ذخیره ی مقدارهای متغیر را محاسبه کرده وبه آن اختصاص می دهد. سرانجام، آن مقدار حافظه ای که در دسترس قرارمی گیرد به اندازه ی 64 bit نخواهد بود. اما چنانچه اصرار دارید، compiler حتماً همان 64 bit را اختصاص دهد (رزرو کند)، حین تخصیص مقدار به متغیر، پسوند L را به آن اضافه کنید. به مثال زیر توجه کنید.

```
class Exercise
{
  static void Main()
  {
    long countryArea;
    countryArea = 5638648L;
    System.Console.Write("Country Area: ");
    System.Console.Write(countryArea);
    System.Console.Write("km2\n");
    System.Console.ReadKey();
  }
}
```

پس به خاطر داشته باشید که نوع داده های short، int، uint یا ushort همگی در یک متغیر long جای می گیرند.

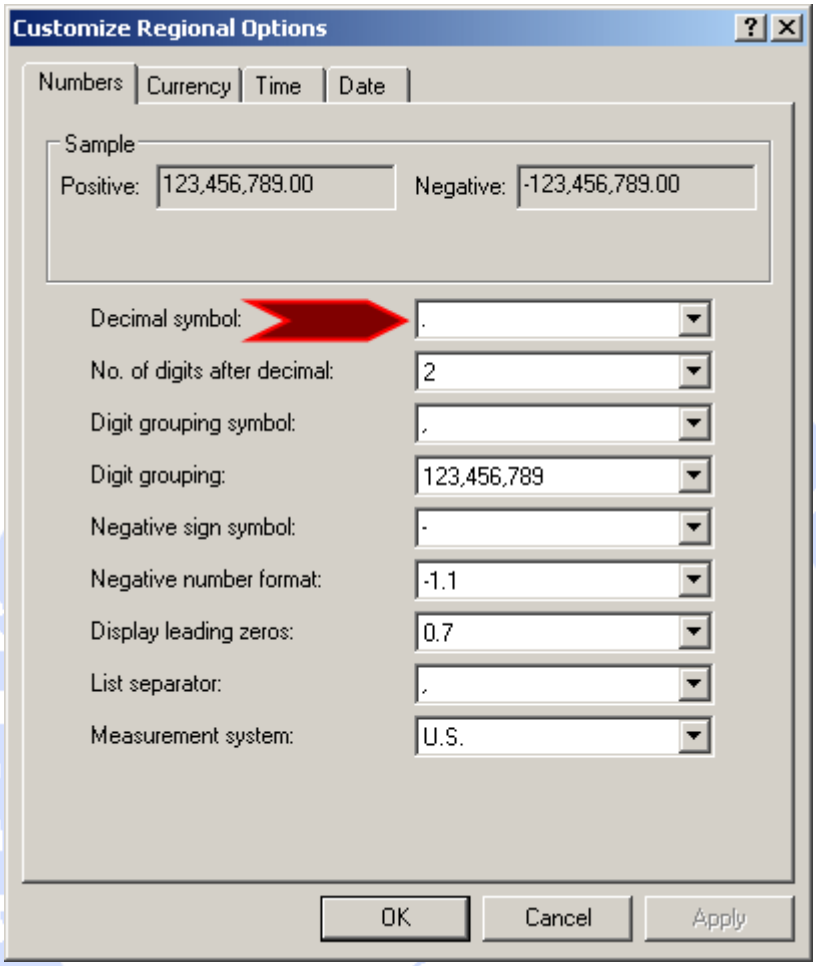
Unsigned long integers

می توان برای ذخیره سازی integer های مثبت و منفی از ترکیب 64 bit ای استفاده کرد. در برخی موارد، برای ذخیره سازی اعدادی که فقط مثبت هستند ولی بزرگ، به متغیر نیاز پیدا می کنید. به منظور معرفی چنین متغیری، می توان از نوع داده ی ulong کمک گرفت. متغیری که به عنوان ulong تعریف می شود، می تواند اعدادی را که از 0 تا 18، 446، 744، 073، 709، 551، 615 متغیر است را در 64 bit جای دهد.

اعداد حقیقی (real numbers)

عدد حقیقی، عددی است که یک بخش اعشاری (decimal part) را نشان می دهد. یک عدد می تواند به وسیله ی یک علامت به دو بخش تقسیم شده باشد، که از آن علامت با نام علامت اعشار (decimal separator/symbol) یاد می شود. البته، این علامت از کشوری به کشور دیگر یا زبانی به زبان دیگر متفاوت است. برای مثال، در زبان انگلیسی آمریکایی این علامت به صورت نقطه (.) نمایش داده می شود.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



در هر دو طرف این علامت اعداد مقدار یک رقم را مشخص می کنند. اعدادی که در سمت راست این علامت قرار می گیرند، در واقع میزان دقت رقم مورد نظر را نشان می دهند.

اعداد ممیز شناور (floating-point numbers)

Integer هایی که تاکنون با آن ها برخورد داشتیم، مقدارهای اعشاری (decimal) را نشان نمی دادند. C# برای این منظور، مقدارهای شناور (floating values) را برای برنامه نویس فراهم می کند. به منظور تعریف اصلی ترین مقدار شناور، از کلیدواژه ی float استفاده می کنیم. متغیری که با float معرفی می شود، قادر است اعداد حقیقی ای که در رنج Ã‚Â±1.5 xÃ‚Â10â^'45 تا Ã‚Â±3.4 xÃ‚Â1038 قرار می گیرند را با ۷ رقم اعشار در 32 bit ذخیره کند. مثال آن را در زیر مشاهده می کنید.

```
class Exercise
{
  static void Main()
  {
```

```
    float distance;
  }
}
```

اعداد با دو رقم اعشار

چنانچه متغیری بزرگتر از ظرفیت float بود (به اندازه ای بزرگ بود که float قادر به نشان دادن آن نیست) و همچنین به دقت اعشار بیشتری نیاز داشت، باید متغیر نام برده را با کلید واژه های var یا double معرفی کرد. به مثال زیر توجه کنید.

```
class Exercise
{
  static void Main()
  {
    var number = 62834.9023;
    System.Console.Write("Number: ");
    System.Console.WriteLine(number);
    System.Console.ReadKey();
  }
}
```

نتیجه ی زیر حاصل می گردد.

```
Number: 62834.9023
Press any key to continue...
```

متغیری که با کلیدواژه ی double معرفی می شود، ارقامی که در رنج Ã‚Â±5.0 x 10â^'324 تا Ã‚Â±1.7 x Ã‚Â10308 قرار می گیرند را با ۱۵ یا ۱۶ رقم اعشار و در 64 bit ذخیره می کند. نوع داده ی double نتایج دقیق تری را نسبت به float یا var به دست می دهد، به خصوص در ارتباط با ذخیره سازی اعداد با رقم اعشار طولانی. اما چنانچه اصرار دارید، حتماً از کلیدواژه ی float برای معرفی متغیر استفاده شود، هنگام تخصیص مقدار به متغیر نام برده، پسوند f یا F را به مقدار اضافه کنید.

```
class Exercise
{
  static void Main()
  {
    float distance;
    distance = 248.38F;
    System.Console.Write("Distance = ");
    System.Console.Write(distance);
    System.Console.WriteLine("km\n");
    System.Console.ReadKey();
  }
}
```

نتییجه زیر حاصل می گردد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
Distance = 248.38km
```

به خاطر داشته باشید، هنگامی که متغیری با var معرفی می شود (و مایلید که مقدار یک رقم اعشار داشته باشد)، باید پسوند f یا F را به مقدار بالا اضافه کنید.

مثال

```
class Exercise
{
  static void Main()
  {
    var number = 62834.9023F;
    System.Console.Write("Number: ");
    System.Console.WriteLine(number);
    System.Console.ReadKey();
  }
}
```

حال، چنانچه مایلید مقداری (value) با دو رقم اعشار شناخته شود، باید پسوند d یا D را به آن ضمیمه کنید. مثال

```
class Exercise
{
  static void Main()
  {
    var number = 62834.9023D;
    System.Console.Write("Number: ");
    System.Console.WriteLine(number);
    System.Console.ReadKey();
  }
}
```

به کاربردن متغیری با دو رقم اعشار

به منظور استفاده از مقداری با دو رقم اعشار، فایل را به صورت زیر تغییر دهید.

```
class Order
{
  static void Main()
  {
    byte? shirts = null;
    byte? pants = null;
    ushort? otherItems = null;
    uint? orderDay = null;
    uint? orderMonth = null;
    uint? orderYear = null;
    double? mondayDiscount = null;
```

```
        shirts = 4;
        pants = 0;
        otherItems = 3;
        orderDay = 15;
        orderMonth = 7;
        orderYear = 2002;
        mondayDiscount = 0.25D; // 25%
        System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
        System.Console.WriteLine("=========================");
        System.Console.Write("Order Date: ");
        System.Console.Write(orderMonth);
        System.Console.Write('/');
        System.Console.Write(orderDay);
        System.Console.Write('/');
        System.Console.WriteLine(orderYear);
        System.Console.WriteLine("-------------------------");
        System.Console.WriteLine("Item Type  Qty");
        System.Console.WriteLine("-------------------------");
        System.Console.Write("Shirts    ");
        System.Console.WriteLine(shirts);
        System.Console.Write("Pants     ");
        System.Console.WriteLine(pants);
        System.Console.Write("Other Items ");
        System.Console.WriteLine(otherItems);
        System.Console.WriteLine("-------------------------");
        System.Console.Write("Monday Discount: ");
        System.Console.Write(mondayDiscount);
        System.Console.WriteLine('%');
        System.Console.WriteLine("=========================");
        System.Console.ReadKey();
    }
}
```

۲. برای مشاهده ی نتیجه، برنامه را اجرا کنید.

```
-/- Georgetown Cleaning Services -/-
    =========================
    Order Date: 7/15/2002
    -------------------------
    Item Type  Qty
    -------------------------
    Shirts    4
    Pants     0
    Other Items 3
    -------------------------
    Monday Discount: 0.25%
    =========================
```

۳. پنجره ی DOS را ببندید.

Decimal

از نوع داده ی decimal، می توان برای تعریف متغیری که قادر است مقادیر بسیار بزرگ (که برای ذخیره به ترکیب 128 bit نیاز دارند) را در خود جای دهد، استفاده کرد. برای این منظور از کلیدواژه ی decimal استفاده می شود. مقادیری که در متغیر دهدهی (decimal) ذخیره می شود، از Â±1.0 xÂ10â^'28 تا Â±7.9 xÂ1028 متغیر بوده و دقت اعشاری از ۲۸ تا ۲۹ رقم دارند. به خاطر این دقت بسیار بالا، از نوع داده ی decimal به منظور نشان دادن ارزش پول استفاده می شود.

پس از تعریف متغیر دهدهی (decimal)، می توان آن را با یک عدد طبیعی مقداردهی اولیه کرد. برای این منظور، هنگام مقدار دهی اولیه، پسوند m یا M را به مقدار مذکور ضمیمه کنید.

مثال

```
class Exercise
{
  static void Main()
  {
    decimal hourlySalary;
    hourlySalary = 24.25M;
    System.Console.Write("Hourly Salary = ");
    System.Console.WriteLine(hourlySalary);
    System.Console.WriteLine();
    System.Console.ReadKey();
  }
}
```

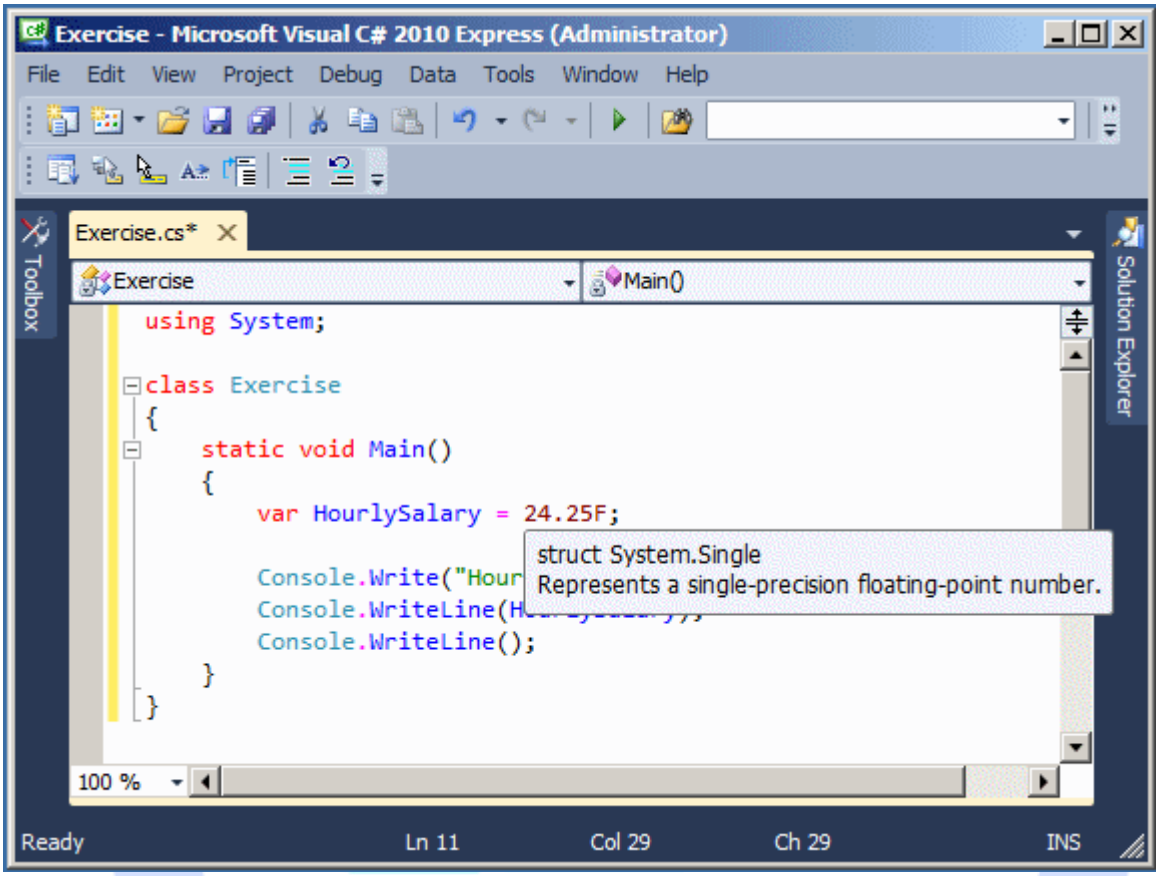
نتیجه ی زیر به دست می آید.

```
Hourly Salary = 24
```

همان طور که پیش تر بیان شد، هنگام تعریف یا مقداردهی اولیه ی متغیر حقیقی، آن پسوندی که به مقدار ضمیمه می کنید به compiler می فهماند چه نوع مقدار و چه میزان حافظه برای ذخیره سازی متغیر نام برده باید تخصیص داده شود.

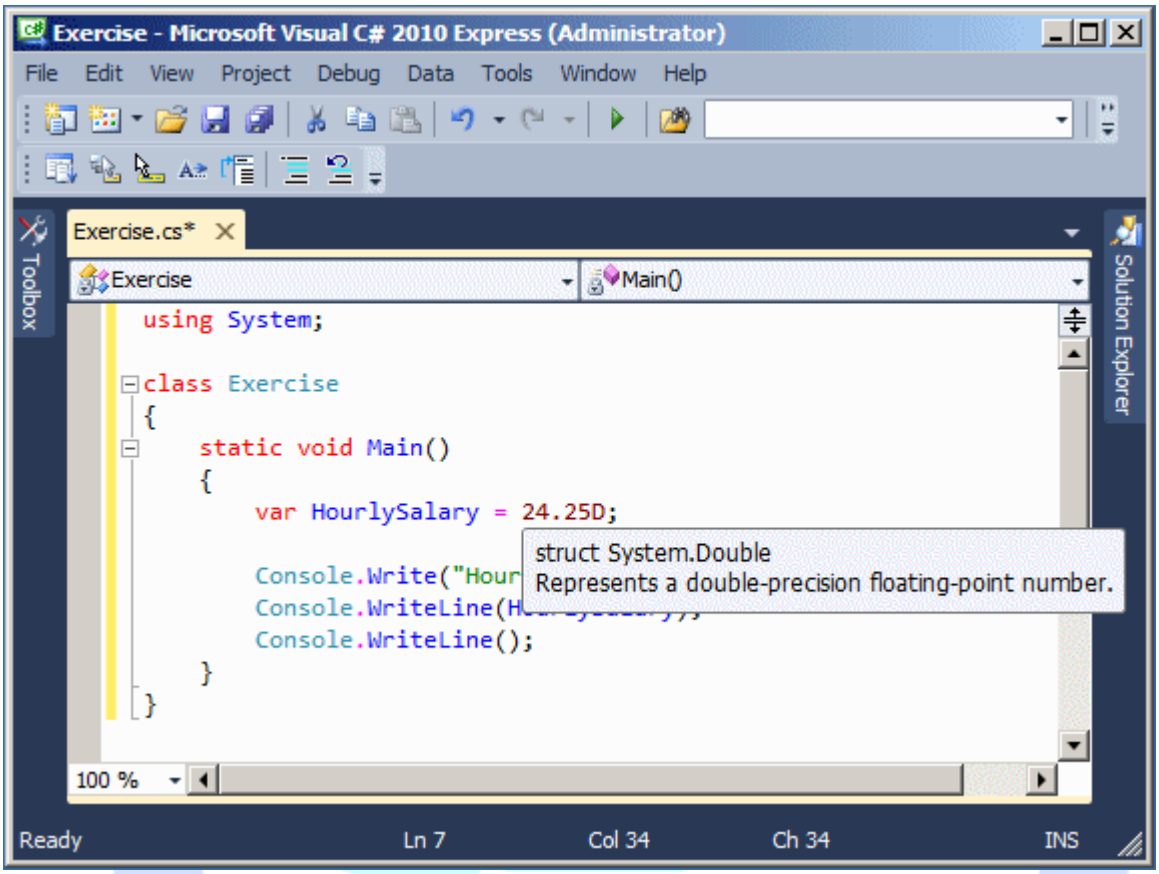
چنانچه مقداری پسوند f یا F را به یدک بکشد، floating point number (عدد ممیز اعشار) با تنها یک رقم اعشار محسوب می شود.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



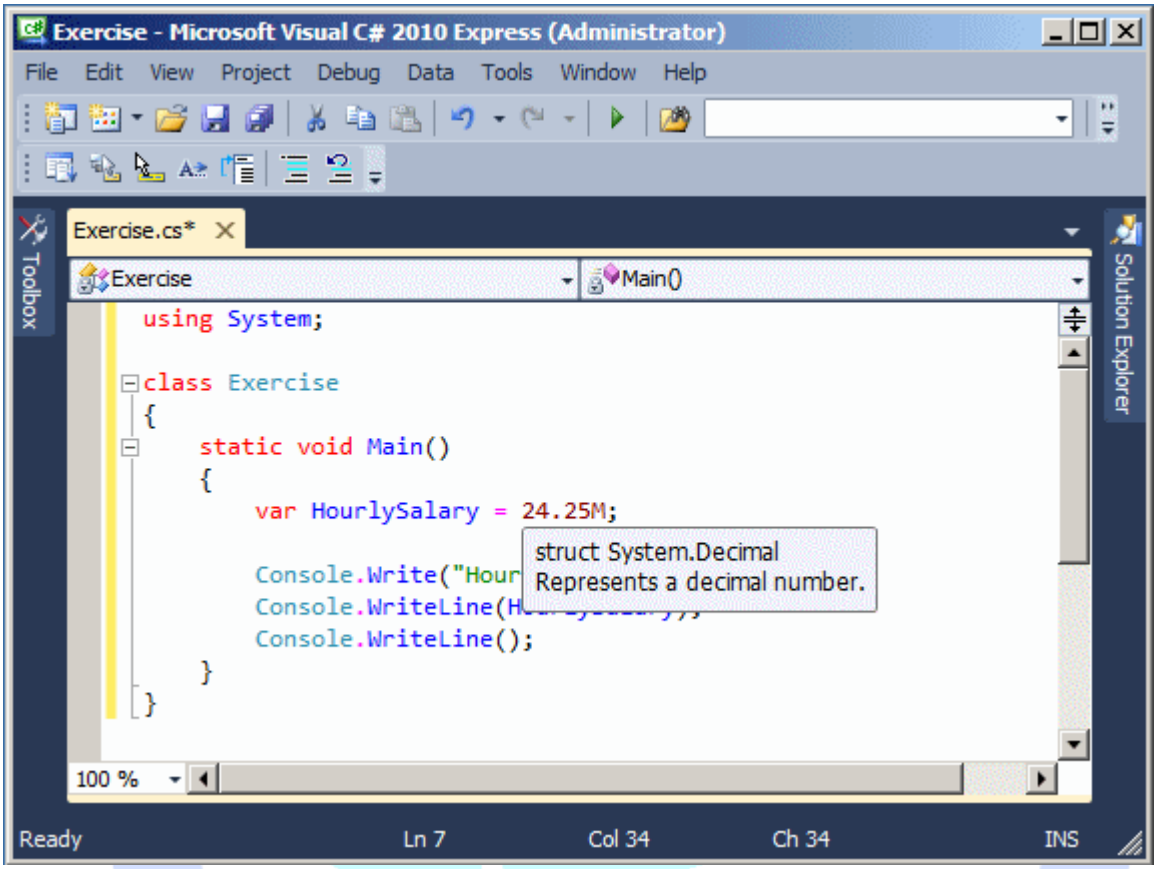
چنانچه مقداری پسوند d یا D را به یدک بکشد، floating point number (عدد ممیز اعشار) با دو رقم اعشار تلقی می گردد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



چنانچه مقداری پسوند m یا M را داشت، یک عدد اعشاری بزرگ (large decimal number) اطلاق می شود

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



به برنامه زیر توجه کنید.

```
class Program
{
  static void Main()
  {
    System.Console.Write("560 / 672 = ");
    System.Console.WriteLine(560 / 672);
    System.Console.ReadKey();
  }
}
```

هدف این برنامه به دست آوردن نتیجه ی تقسیم ۵۶۰ بر ۶۷۲ است.

```
560 / 672 = 0
Press any key to continue…
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

همان طور که مشاهده می کنید، هنگامی که چنین ارقامی به برنامه داده می شوند، compiler آن ها را به عنوان integer می شناسد و نتیجه ی ۰ را به دست می دهد. حال، یک روش برای تصحیح این معادله، اضافه کردن جایی برای اعشار به حداقل یکی از ارقام است.

مثال

```
class Program
{
  static void Main()
  {
    System.Console.Write("560 / 672 = ");
    System.Console.WriteLine(560.00 / 672.00);
    System.Console.ReadKey();
  }
}
```

این بار، compiler مقدار را به عنوان floating-point number (عدد با ممیز شناور) می شناسد، اما کدام ؟ به صورت پیش فرض، compiler نوع داده ی double را به کار می برد. برنامه نتیجه ی زیر را تولید می کند.

```
560 / 672 = 0.833333333333333
Press any key to continue…
```

به منظور تعیین نوع مقدار داده شده (float، double یا decimal) پسوند مربوط را به آن اضافه کنید.

```
class Program
{
  static void Main()
  {
    System.Console.Write("560 / 672 = ");
    System.Console.WriteLine(560F / 672f);
    System.Console.ReadKey();
  }
}
```

این بار برنامه نتیجه ی زیر را می دهد.

```
560 / 672 = 0.8333333
Press any key to continue…
```

به کاربردن مقادیر decimal

به منظور استفاده از متغیرهای decimal، فایل را به صورت زیر اصلاح کنید.

```
class Order
{
```

```
static void Main()
{
  byte? shirts = null;
  decimal? priceOneShirt = null;
  byte? pants = null;
  decimal? priceAPairOfPants = null;
  ushort? otherItems = null;
  decimal? priceOtherItems = null;
  uint? orderDay = null;
  uint? orderMonth = null;
  uint? orderYear = null;
  double? mondayDiscount = null;
  shirts = 5;
  priceOneShirt = 0.95M;
  pants = 2;
  priceAPairOfPants = 1.95M;
  otherItems = 3;
  priceOtherItems = 4.55M;
  orderDay = 15;
  orderMonth = 7;
  orderYear = 2002;
  mondayDiscount = 0.25D; // 25%
  System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
  System.Console.WriteLine("==========================");
  System.Console.Write("Order Date: ");
  System.Console.Write(orderMonth);
  System.Console.Write('/');
  System.Console.Write(orderDay);
  System.Console.Write('/');
  System.Console.WriteLine(orderYear);
  System.Console.WriteLine("------------------------");
  System.Console.WriteLine("Item Type  Qty Unit Price");
  System.Console.WriteLine("------------------------");
  System.Console.Write("Shirts    ");
  System.Console.Write(shirts);
  System.Console.Write("    ");
  System.Console.WriteLine(priceOneShirt);
  System.Console.Write("Pants     ");
  System.Console.Write(pants);
  System.Console.Write("    ");
  System.Console.WriteLine(priceAPairOfPants);
  System.Console.Write("Other Items ");
  System.Console.Write(otherItems);
  System.Console.Write("    ");
  System.Console.WriteLine(priceOtherItems);
  System.Console.WriteLine("------------------------");
  System.Console.Write("Monday Discount: ");
  System.Console.Write(mondayDiscount);
  System.Console.WriteLine('%');
  System.Console.WriteLine("==========================");
```

```
    System.Console.ReadKey();
  }
}
```

۲. برنامه را اجرا کنید. نتیجه ی زیر به دست می آید.

```
   -/- Georgetown Cleaning Services -/-
  ==========================
  Order Date: 7/15/2002
  -------------------------
  Item Type  Qty Unit Price
  -------------------------
  Shirts     5    0.95
  Pants      2    1.95
  Other Items 3    4.55
  -------------------------
  Monday Discount: 0.25%
  ==========================
```

۳. پنجره ی DOS را ببندید.

نوع داده های جانبی (Accessory Data Type)

رشته ها (Strings)

رشته یک فضای خالی، یک کاراکتر، یک کلمه یا یک گروه کلمه است که compiler باید آن را " همان طور که هست " در نظر بگیرد، یعنی نباید به این که رشته از چه چیزی تشکیل شده زیاد دقت کند، مگر این که شما مستقیم غیر این را به compiler دستور دهید. به عبارت دقیق تر، می توانید هر چه دلخواهتان است در یک رشته بگنجانید.

اصولاً، مقدار یک رشته با علامت نقل و قول (") آغاز می شود و به آن نیز ختم می شود. نمونه ی یک رشته به این صورت می باشد : "Welcome to the World of C# Programming!". برای این که یک رشته در صفحه ی کنسول نمایش داده شود، می توان آن را در دستور System.Console.Write() قرار داد.

```
class Exercise
 {
   static void Main()
   {
```

```
    System.Console.WriteLine("Welcome to the World of C# Programming!");
    System.Console.ReadKey();
  }
}
```

نتیجه ی زیر به دست می آید.

```
Welcome to the World of C# Programming!
Press any key to continue...
```

ممکن است مجبور به استفاده از رشته ای شوید که مقدار آن از پیش برای شما شناس نباشد. در این مورد، باید یک متغیر رشته (string variable) تعریف کرد. برای این منظور، از کلیدواژه های var یا string استفاده می شود و به دنبال آن اسم متغیر مورد نظر قرار داده می شود. می توان یک رشته را با یک فضای خالی، یک کاراکتر، یک نشانه، یک کلمه ، یک گروه کلمه یا حتی null مقداردهی اولیه (initialize) کرد (به هیچ وجه نباید از string? استفاده کرد). مقدار (value) ای که به رشته داده می شود باید حتماً داخل علامت (" ") قرار گیرد. (مگر این که آن مقدار null) باشد. دو نمونه زیر را مشاهده کنید.

```
class Exercise
{
  static void Main()
  {
    var team = "Real Madrid";
    string country = "GuinÃ©e Equatoriale";
    System.Console.WriteLine("Welcome to the World of C# Programming!");
    System.Console.Write("Team:  ");
    System.Console.WriteLine(team);
    System.Console.Write("Country: ");
    System.Console.WriteLine(country);
    System.Console.WriteLine();
    System.Console.ReadKey();
  }
}
```

نتیجه

```
Welcome to the World of C# Programming!
Team:  Real Madrid
Country: GuinÃ©e Equatoriale
Press any key to continue...
```

به کار بردن رشته ها

۱. برای استفاده از رشته (string)، فایل را به صورت زیر تغییر دهید.

```
class Order
{
  static void Main()
  {
    string customerName;
    string customerHomePhone;
    byte? shirts = null;
    decimal? priceOneShirt = null;
    byte? pants = null;
    decimal? priceAPairOfPants = null;
    ushort? otherItems = null;
    decimal? priceOtherItems = null;
    uint? orderDay = null;
    uint? orderMonth = null;
    uint? orderYear = null;
    double? mondayDiscount = null;
    customerName = "Gregory Almas";
    customerHomePhone = "(301) 723-4425";
    shirts = 5;
    priceOneShirt = 0.95M;
    pants = 2;
    priceAPairOfPants = 1.95M;
    otherItems = 3;
    priceOtherItems = 4.55M;
    orderDay = 15;
    orderMonth = 7;
    orderYear = 2002;
    mondayDiscount = 0.25D; // 25%
    System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
    System.Console.WriteLine("=========================");
    System.Console.Write("Customer: ");
    System.Console.WriteLine(customerName);
    System.Console.Write("Home Phone: ");
    System.Console.WriteLine(customerHomePhone);
    System.Console.Write("Order Date: ");
    System.Console.Write(orderMonth);
    System.Console.Write('/');
    System.Console.Write(orderDay);
    System.Console.Write('/');
    System.Console.WriteLine(orderYear);
    System.Console.WriteLine("-------------------------");
    System.Console.WriteLine("Item Type  Qty Unit Price");
    System.Console.WriteLine("-------------------------");
    System.Console.Write("Shirts     ");
```

```
        System.Console.Write(shirts);
        System.Console.Write("    ");
        System.Console.WriteLine(priceOneShirt);
        System.Console.Write("Pants     ");
        System.Console.Write(pants);
        System.Console.Write("    ");
        System.Console.WriteLine(priceAPairOfPants);
        System.Console.Write("Other Items ");
        System.Console.Write(otherItems);
        System.Console.Write("    ");
        System.Console.WriteLine(priceOtherItems);
        System.Console.WriteLine("------------------------");
        System.Console.Write("Monday Discount: ");
        System.Console.Write(mondayDiscount);
        System.Console.WriteLine('%');
        System.Console.WriteLine("==========================");
        System.Console.ReadKey();
    }
}
```

۲. حال، برنامه را اجرا کنید. نتیجه ی زیر حاصل می گردد.

```
-/- Georgetown Cleaning Services -/-
    ==========================
    Customer:  Gregory Almas
    Home Phone: (301) 723-4425
    Order Date: 7/15/2002
    ------------------------
    Item Type  Qty Unit Price
    ------------------------
    Shirts     5    0.95
    Pants      2    1.95
    Other Items 3    4.55
    ------------------------
    Monday Discount: 0.25%
    ==========================
```

۳. اکنون می توانید پنجره ی DOS را ببندید.

تاریخ و زمان

تاریخ در واقع واحدی برای سنجش تعداد روزها، ماه ها، سال هایی است که در یک برهه ی مشخصی از زمان سپری شده. زمان نیز واحدی برای شمارش تعداد ثانیه هایی است که از نیمه شب روز معین تا کنون سپری شده. به منظور تعریف متغیری ویژه ی زمان و تاریخ، باید از نوع داده ی DateTime بهره گرفت.

```
class Exercise
{
  static void Main()
  {
    DateTime DateHired;
  }
}
```

.NET Framework. تاریخ شروع یک دوره را به January 1، 0001، نیمه شب (0:00 AM یا 12:00:00) تنظیم می کند. چنانچه مقدار مشخصی به آن اختصاص داده نشده باشد، متغیر با 1/1/0001 نیمه شب، مقدار دهی اولیه می شود.

شی ها (Objects)

نوع داده ی Object به منظور تعریف متغیری به کار می رود که نوع آن از پیش تعیین نشده و ممکن است نوع آن هر یک از نوع داده هایی که در این مبحث تعریف کردیم باشد. پس از تعریف متغیر شی (object variable)، می توان مقدار آن را هر گونه که مایلید به کاربردید. برای مثال، می توان متغیر موردنظر را داخل پرانتزهای System.Console.Write() یا System.Console.WriteLine() قرار داد تا در صفحه ی کنسول به نمایش گذاشته شود.

```
class Exercise
{
  static void Main()
  {
    var EmployeeName = "Ernestine Lamb";
    object Address = "10244 Lockwood Drive";
    System.Console.Write("Employee Name: ");
    System.Console.WriteLine(EmployeeName);
    System.Console.Write("Home Address: ");
    System.Console.WriteLine(Address);
    System.Console.WriteLine();
    System.Console.ReadKey();
  }
}
```

نتیجه ی زیر به دست می آید.

```
Employee Name: Ernestine Lamb
Home Address: 10244 Lockwood Drive
Press any key to continue...
```

متغیری که با نوع داده ی object تعریف شود، قادر است هر مقداری را در خود جای دهد.

```
class Program
{
  static void Main()
  {
    object propertyNumber = "293749";
    object propertyType = 'S';
    object stories = 3;
    object bedrooms = 4;
    object value = 425880;
    System.Console.WriteLine("=//= Altair Realtors =//=");
    System.Console.WriteLine("Properties Inventory");
    System.Console.Write("Property #:  ");
    System.Console.WriteLine(propertyNumber);
    System.Console.Write("Property Type: ");
    System.Console.WriteLine(propertyType);
    System.Console.Write("Stories:    ");
    System.Console.WriteLine(stories);
    System.Console.Write("Bedrooms:    ");
    System.Console.WriteLine(bedrooms);
    System.Console.Write("Market Value: ");
    System.Console.WriteLine(value);
    System.Console.ReadKey();
  }
}
```

نتیجه ی زیر به دست می آید.

```
=//= Altair Realtors =//=
Properties Inventory
Property #:  293749
Property Type: S
Stories:    3
Bedrooms:    4
Market Value: 425880
Press any key to continue...
```

ثابت ها (constants)

فرض کنید باید رقمی همچون 39.37 را بارها و بارها استفاده کنید.

مثال

```
class Exercise
{
  static void Main()
  {
    double meter, inch;
    meter = 12.52D;
    inch = meter * 39.37D;
    System.Console.Write(meter);
    System.Console.Write("m = ");
    System.Console.Write(inch);
    System.Console.WriteLine("in\n");
    System.Console.ReadKey();
  }
}
```

نمونه ای از برنامه ی در حال اجرا را در زیر مشاهده می کنید.

```
12.52m = 492.9124in
```

در صورت استفاده مکرر از این رقم، ممکن است اشتباهی کرده و به جای ۳۹٫۳۷ تایپ کنید 3937 یا 3.937. برنامه ی زیر را در نظر بگیرد.

```
class Exercise
{
  static void Main()
  {
    double meter, inch;
    meter = 12.52D;
    inch = meter * 39.37;
    System.Console.Write(meter);
    System.Console.Write("m = ");
    System.Console.Write(inch);
    System.Console.WriteLine("in\n");
    meter = 12.52D;
    inch = meter * 3.937;
    System.Console.Write(meter);
    System.Console.Write("m = ");
    System.Console.Write(inch);
    System.Console.WriteLine("in\n");
    meter = 12.52D;
    inch = meter * 393.7;
    System.Console.Write(meter);
    System.Console.Write("m = ");
    System.Console.Write(inch);
    System.Console.WriteLine("in\n");
    System.Console.ReadKey();
  }
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
}
```

نتیجه ی زیر حاصل می گردد.

```
۱۲/۵۲m = 492.9124in
۱۲/۵۲m = 49.29124in
۱۲/۵۲m = 4929.124in
```

به خاطر اشتباهاتی از این دست، همان محاسبه ممکن است نتایجی کاملاً متفاوت تولید کند. به منظور جلوگیری از خطاهایی از این دسته، می توان از یک متغیر (به جای ثابت) برای ذخیره کردن مقدار استفاده کرد. حال، در صورت نیاز می توان به جای خود مقدار، به متغیر آن به راحتی دسترسی پیدا کرد. لازم به ذکر است که رقمی همچون 39.37 یک ثابت (constant) است.

ثابت مقداری است که هیچگاه تغییر نمی کند، مانند : 244، "ASEC Mimosa"،39.37 یا حتی True. می توان یک متغیر (variable) را طوری تعریف کرد و به کاربرد که مقدار آن مانند ثابت (constant) تغییر نکند.

به منظور ایجاد یک ثابت، باید کلیدواژه ی const را در سمت چپ آن تایپ کنید. به خاطر داشته باشید که حین تعریف ثابت، باید مقدار مناسب به آن اختصاص داد.

مثال

```
const double ConversionFactor = 39.37D;
```

پس از ایجاد ثابت مورد نظر و مقداردهی اولیه ی مناسب آن، می توان اسم آن را در جایی که ثابت دلخواه باید قرار می گرفت وارد کرد. در زیر نمونه ای از متغیر ثابت (constant variable)، که بارها استفاده شده مشاهده می کنید.

```
class Exercise
{
  static void Main()
  {
    const double ConversionFactor = 39.37D;
    double meter, inch;
    meter = 12.52D;
    inch = meter * ConversionFactor;
    System.Console.Write(meter);
    System.Console.Write("m = ");
    System.Console.Write(inch);
    System.Console.WriteLine("in\n");
    meter = 12.52D;
    inch = meter * ConversionFactor;
    System.Console.Write(meter);
```

```
        System.Console.Write("m = ");
        System.Console.Write(inch);
        System.Console.WriteLine("in\n");
        meter = 12.52D;
        inch = meter * ConversionFactor;
        System.Console.Write(meter);
        System.Console.Write("m = ");
        System.Console.Write(inch);
        System.Console.WriteLine("in\n");
        System.Console.ReadKey();
    }
}
```

نتیجه ی زیر به دست می آید.

```
۱۲/۵۲m = 492.9124in
۱۲/۵۲m = 492.9124in
۱۲/۵۲m = 492.9124in
```

همان طور که مشاهده می کنید، این بار نتیجه دقیق تر درآمده است. همچنین اگر چیزی را اشتباه تایپ کنید، یک پیغام خطا صادر می شود که سر انجام زمان بیشتری در اختیار شما قرار می دهد تا مشکل را برطرف کنید.

به منظور مقداردهی اولیه ی یک متغیر ثابت، مقداری که در سمت راست عملگر جایگزین = قرار می گیرد، باید حتماً یک ثابت یا مقداری باشد که compiler به عنوان ثابت بشناسد. به جای استفاده از یک ثابت شناس، می توان متغیری به آن اختصاص داد که از پیش به عنوان ثابت تعریف شده.

ثابت های توکار

در برنامه ها دو دسته ی اصلی ثابت کاربرد فراوان دارند. می توانید خود یک ثابت خلق کنید (مانند مثال بالا). زبان C# ثابت های مختص خود را دارد. برخی از ثابت ها متعلق به خود زبان C# هستند و برخی دیگر از برنامه ی.NET Framework گرفته می شوند. پیش از به کاربردن یک ثابت، ابتدا باید از وجود ثابت اطمینان کسب کنید. دوم، باید از نحوه ی دسترسی به آن آگاه باشید. ثابتی که بخشی از خود برنامه ی C# است به راحتی از طریق کد قابل دسترسی می باشد (به طور معمول، ثابت های مزبور در فضای اسمی System تعریف می شوند).

Null : این کلیدواژه (Null) در واقع یک ثابت است و برای نشان دادن متغیری استفاده می شود که مقدار مشخص و شناسی ندارد.

PI : ثابتی است که برای نشان دادن نسبت محیط دایره به قطر آن به کار می رود. البته کاربرد این ثابت بیشتر در علم ریاضی است و برای استفاده از آن در برنامه نویسی باید Math.PI را تایپ کنید.

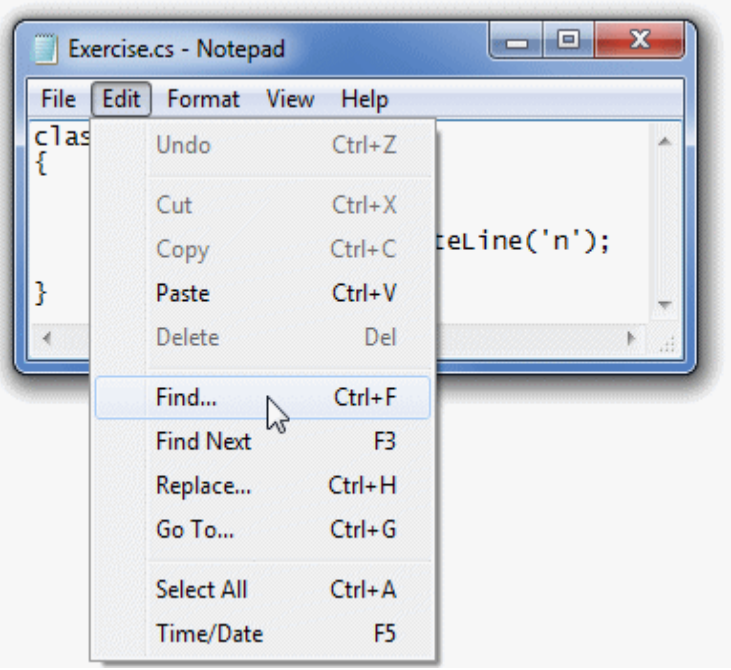
مدیریت کد

ابزار اصلی نوشتن برنامه، نوشتن و وارد کردن کد در داکیومنت است. این کار با نوشتن و وارد کردن کد از طریق صفحه کلید امکان پذیر می باشد. پس از به وجود آوردن کد، عملیات مختلف می توان روی آن انجام داد. معمولاً اولین کاری که باید انجام داد، انتخاب متن است.

دسترسی پیدا کردن به متغیر

همان طور که از قبل گفته شد، کد را می توان در برنامه هایی مثل Notepad یا code editor برنامه ی Microsoft Visual Studio نوشت.

برای پیدا کردن کاراکتر، نشانه، کلمه در یک داکیومنت باید به main menu مراجعه کرده و روی گزینه ی Edit -> Find... کلیک کنید.

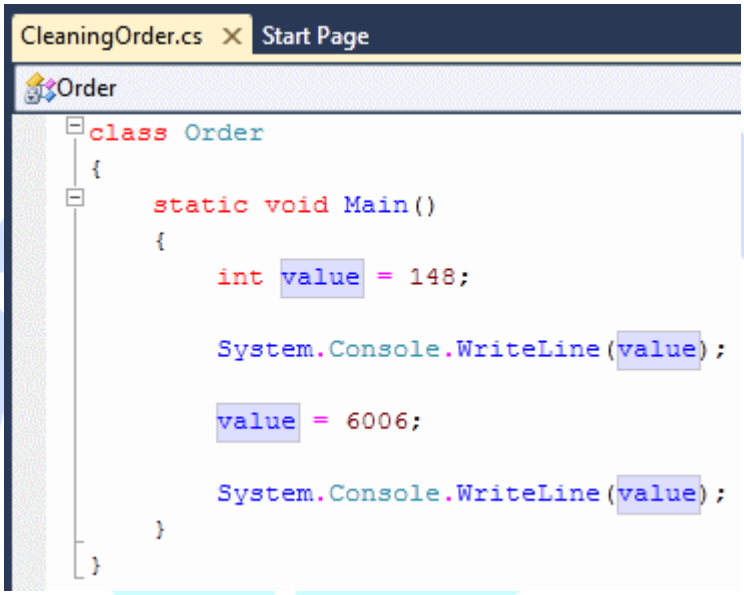


پنجره ی محاوره (dialog box) ای نمایش داده می شود، که می توان در آن نام آیتم موردنظر را وارد کرده، سپس روی گزینه ی Find کلیک کنید. اگر از برنامه ی Microsoft Visual Studio استفاده می کنید برای پیدا کردن تمام کاراکترهای یک متن (منظور یک کاراکتر است که چند بار در یک متن تکرار شده)، ابتدا آیتم مورد نظر را انتخاب کرده، سپس

Main menu را باز کرده، و گزینه ی Edit -> Quick Find را انتخاب کنید.

حال، Ctrl + F را باهم فشار دهید.

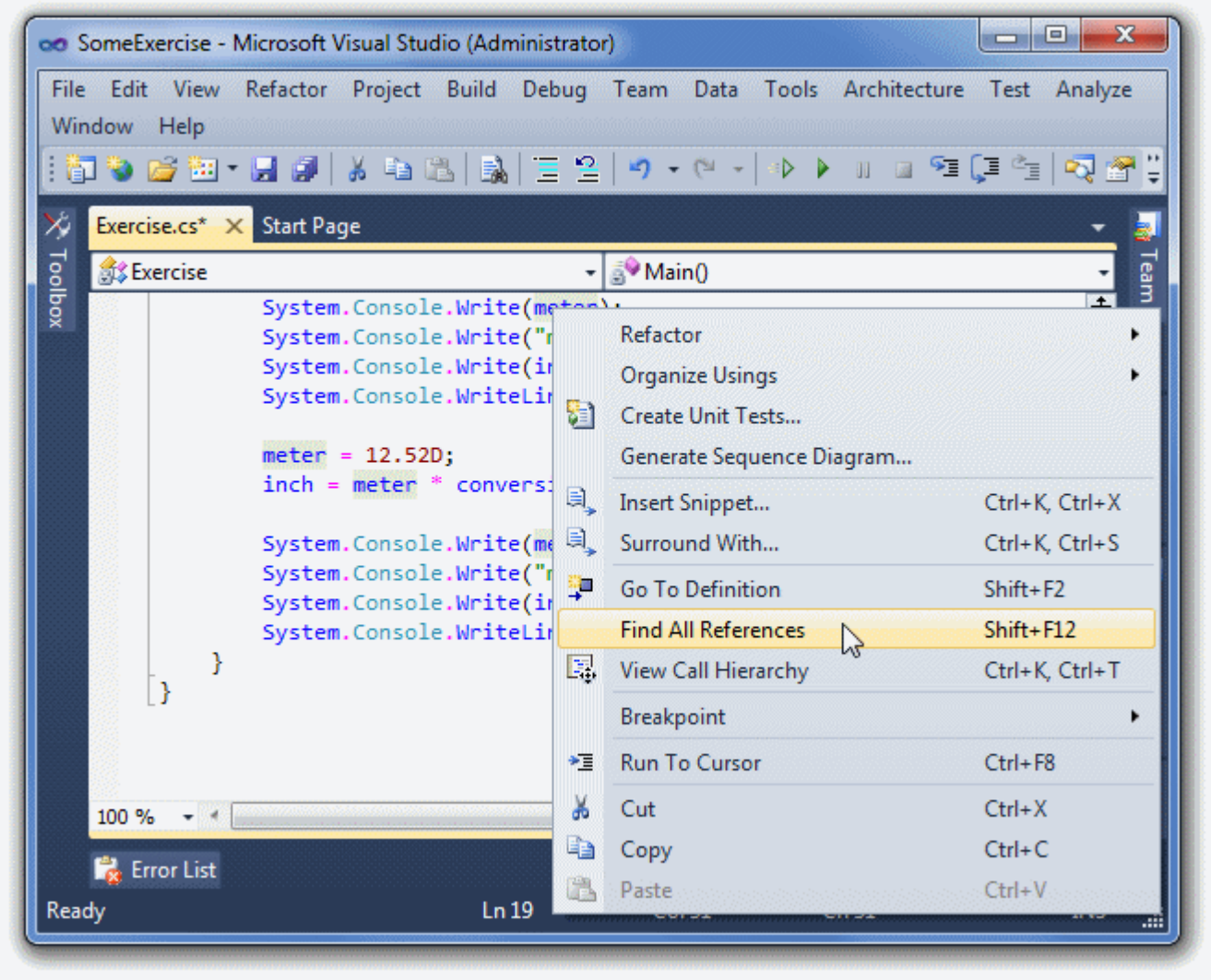
به همین شیوه، چنانچه متغیری دارید که چند بار در یک متن به کار رفته و می خواهید تمام جاهایی که این متغیر در آن ها به کارفته را پیدا کنید، کافی است روی نام آن کلیک کرده و دو ثانیه صبر کنید تا تمام متغیرهای موردنظر یافت شوند (و برجسته شوند).



اکنون، به منظور دریافت فهرستی از تمام مکان هایی که متغیر در آن به کار رفته، در برنامه ی Microsoft Visual Studio دستورات زیر را دنبال کنید :

1. در Code Editor، اسم متغیر را راست کلیک کرده، سپس گزینه ی Find All References را انتخاب کنید.

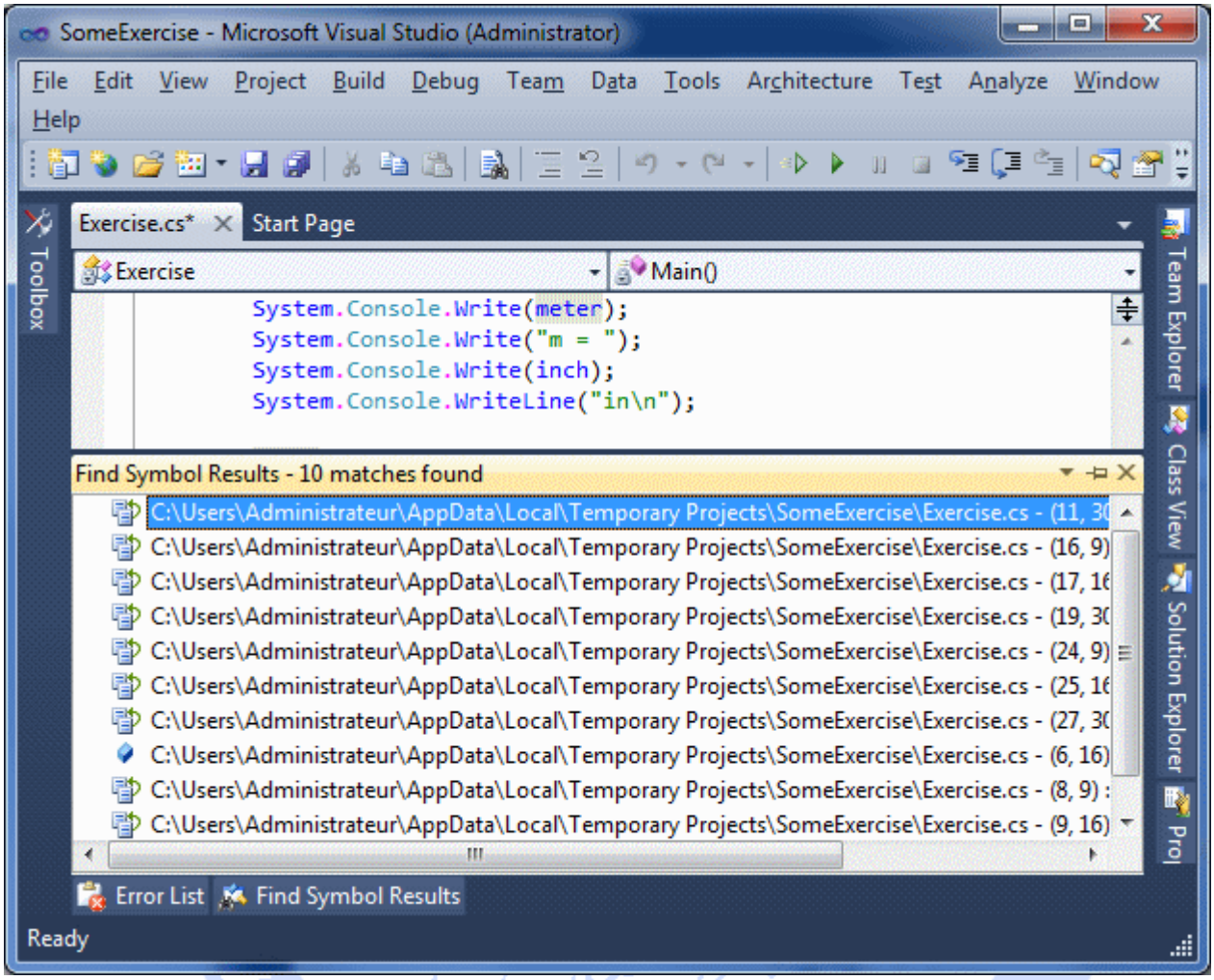
آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



۲. Ctrl + F12، را با هم فشار دهید.

در نتیجه ی این کار، لیستی از تمام بخش هایی که متغیر در آن به کار رفته (در پنجره ی Find) Symbol Results نمایش داده می شود .

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

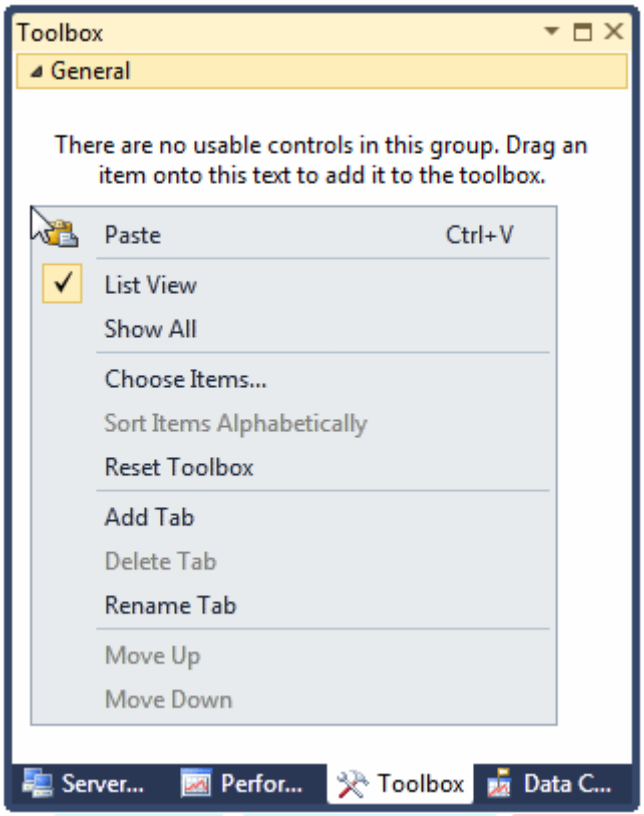


حال، به منظور دستیابی به بخشی که متغیر در آن به کار برده شده، روی ورودی آن دوبار کلیک کنید.

بریدن، کپی کردن و جای گذاری کد

هر شخصی که با کامپیوتر کار کرده، با این عملیات (cut، copy، paste) آشنا است. این عملیات برای ذخیره سازی کد در Microsoft Visual Studio بسیار پرکاربرد هستند. به عبارت دیگر، چنانچه کدی دارید که مایلید در بخش های مختلف استفاده کنید، می توانید آن را در جایی به صورت ذخیره نگه داشته و هر زمان که به آن نیاز پیدا کردید از آن استفاده کنید.

ابتدا، به منظور ذخیره سازی کد، کد را در هر برنامه ی ویرایش متنی مثل word، notepad یا code editor برنامه ی Microsoft Visual Studio تایپ کنید. کد نام برده را انتخاب کرده، سپس آن را روی clipboard کپی کنید. حال، برای حفظ کردن آن، (در Microsoft Visual Studio) Toolbox را باز کنید (main menu را باز کرده، روی View -> Toolbox کلیک کنید). یک فضای خالی را در toolbox انتخاب کرده، روی آن راست کلیک کنید و بعد گزینه ی paste را فشار دهید.



شیوه ی دیگر انجام این کار، این است که کد دلخواه را انتخاب کرده، سپس آن را بکشید و روی صفحه ی Toolbox رها کنید. به همین شیوه، می توانید code item های مختلفی به Toolbox اضافه کنید. پس از جای گذاری کد در Toolbox، کد موردنظر قابل دسترس می شود. حال، به منظور استفاده از کد، آن را از Toolbox گرفته و روی قسمت مورد نظر در code editor قرار دهید.

تغییر اسم متغیر

به منظور تغییر اسم آیتمی در کامپیوتر، ابتدا باید آن را پیدا کرده، سپس آن را ویرایش کنید. چنانچه آن اسم در قسمت های مختلفی به کارفته، همچنان می توان به دنبال همه ی آن ها گشت و آن ها را ویرایش کرد. احتمال خطا کردن در این مورد بسیار بالا است. اگر کد خود را با text editor می نویسید، می توانید تمام نمونه های آن اسم را با گزینه ی Edit -> Replace پیدا کرده و ویرایش کنید. از همین روش می توان در code editor نیز استفاده کرد. متأسفانه این روش تنها برای یک فایل کارگر است. در صورتی که پروژه ی شما فایل های متعددی دارد، واسم ها در آن فایل ها ذخیره شده باشند، این فرایند بسیار دشوار می گردد.

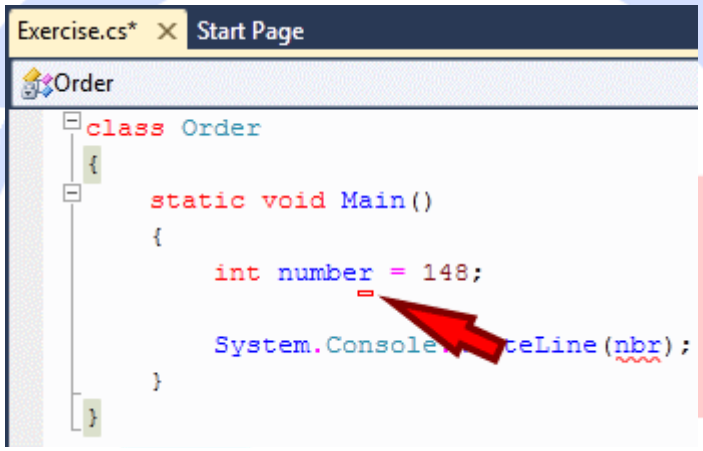
اما برنامه ی Microsoft Visual Studio، فرایند پیدا کردن و تغییر دادن اسم را فوق العاده آسان می سازد. کد زیر را در نظر بگیرید.

```
class Order
```

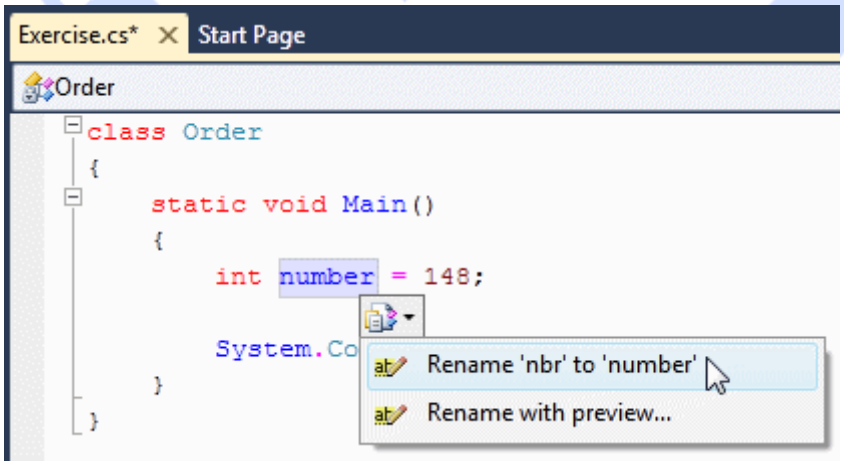
```
{
  static void Main()
  {
    int nbr = 148;
    System.Console.WriteLine(nbr);
  }
}
```

به منظور تغییر اسم یک متغیر، در code editor دستورات زیر را انجام دهید.

روی اسم موردنظر دوبار کلیک کرده، آن را ویرایش کنید. زیر اسم موردنظر یک خط زیرین (_ underline) پدیدار می شود.

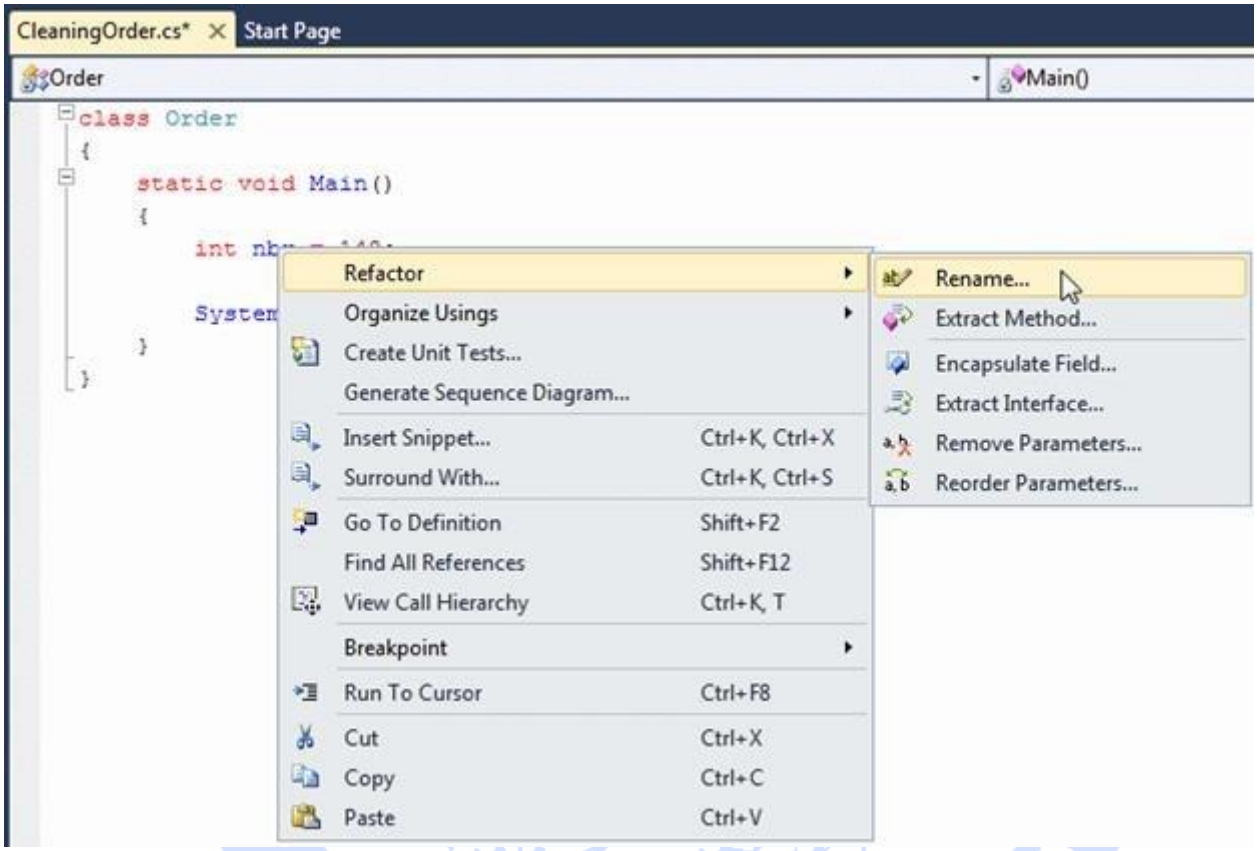


اگر نشان گر موس خود را روی آن قرار دهید، تگی پدیدار می شود که با کلیک کردن روی پیکان (arrow) آن، فهرست گزینه ی کوچکی نمایان می شود.

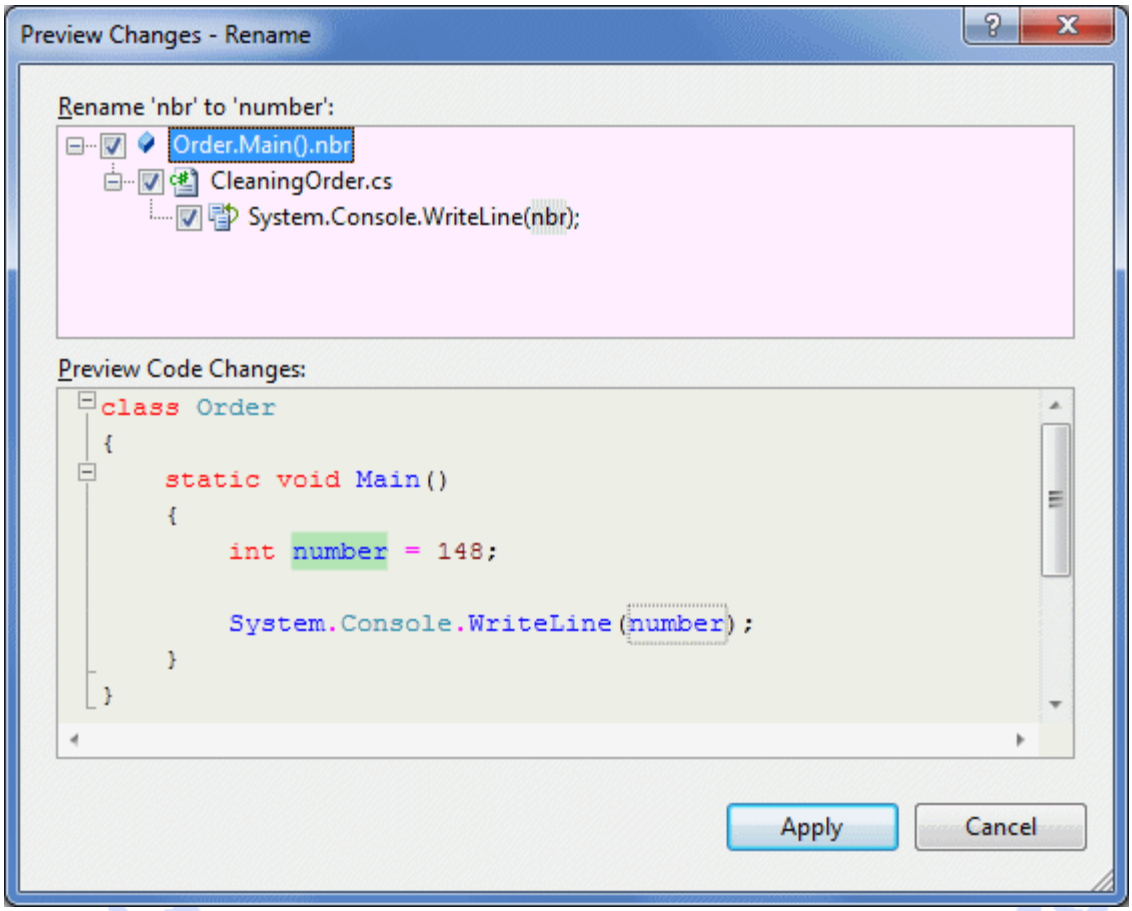


اگر روی گزینه ی اول کلیک کنید، تمام نمونه های آن متغیر عوض می شوند. حال، اگر می خواهید پنجره ی پیش نمایش (next) نمایش داده شود، روی گزینه ی دوم کلیک کنید.

روی آن راست کلیک کرده، نشان گر موس را روی Refactor قرار دهید، سپس گزینه ی rename را کلیک کنید.



اگر می خواهید برنامه ی studio، اسم ها را پیدا کرده و آن ها را در داخل comment ها تغییر دهد، در Comment check box روی گزینه ی Search کلیک کنید. چنانچه، اسم در رشته ها نگه داشته می شود و شما خواهان تغییر آن هستید، در Strings checkbox روی گزینه ی search کلیک کنید. پس از اینکه شما ok را انتخاب کردید، پنجره ی محاوره (dialog box)، Preview Changes-rename روی صفحه نمایش پدیدار می گردد. پنجره ی بالا تمام جاهایی که اسم در آن به کار رفته را برای شما به نمایش می گذارد. حال، می توانید اسم را به دلخواه تغییر دهید.

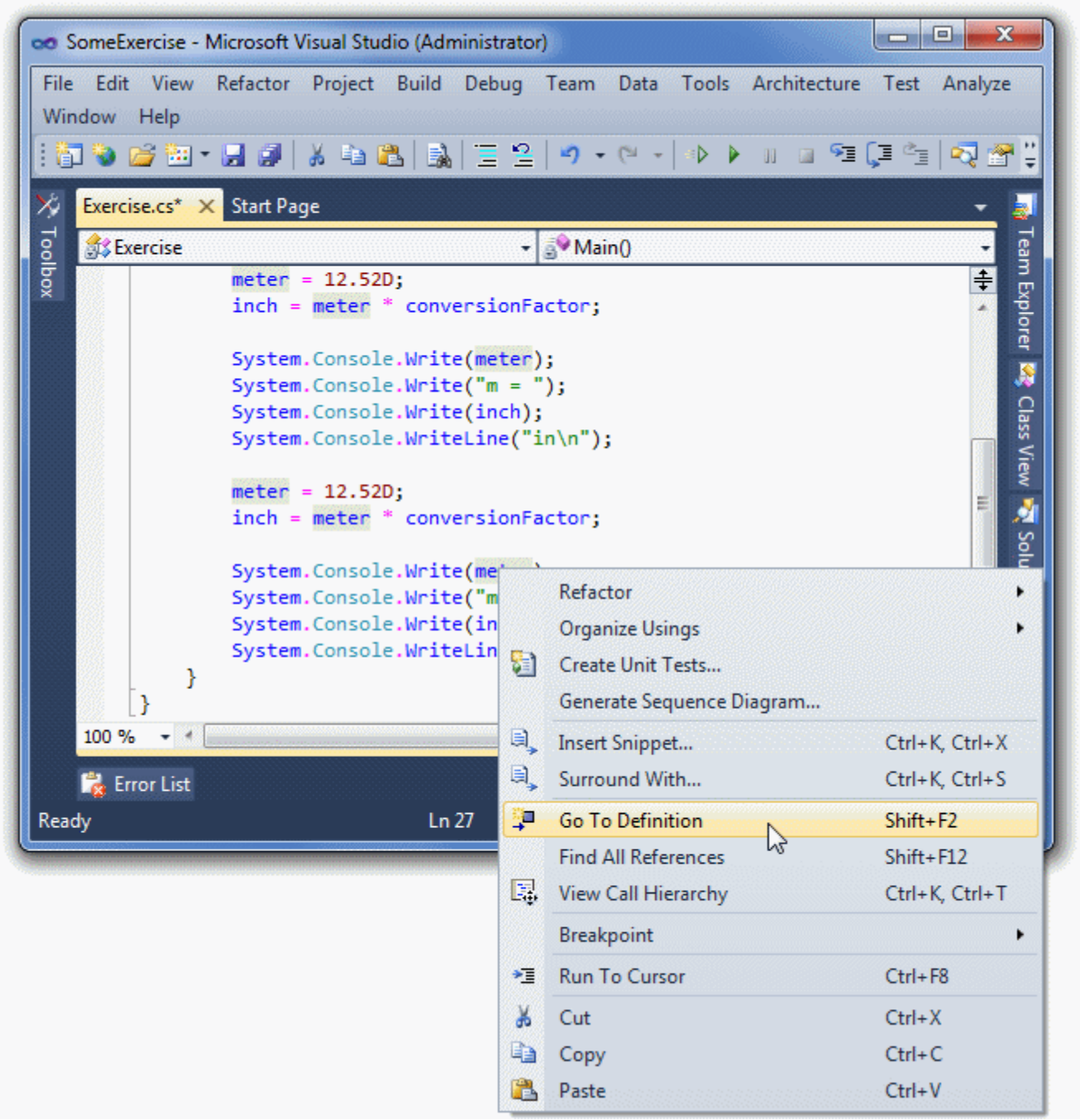


اکنون، به منظور جایگزین کردن اسم دکمه ی Apply را فشار دهید.

دسترسی پیدا کردن به تعریف متغیر (Variable's Declaration)

چنانچه داکیومنتی بسیار طولانی در اختیار دارید که از خطوط بسیار زیادی تشکیل شده، در یک بخش خاص ممکن است با متغیری مواجه شوید که مکانی که متغیر در آن تعریف شده برای شما شناس نیست. حال، در صورت استفاده از Microsoft Visual Studio، به منظور یافتن مکانی که متغیر در آن تعریف شده.

روی متغیر موردنظر کلیک راست کرده، سپس گزینه ی Go to definition را انتخاب کنید.



روی اسم متغیر کلیک کرده و کلید های Shift + F2 را با هم فشار دهید.

در هر دو صورت، برنامه شما را مستقیم به جایی که متغیر در آن تعریف شده می برد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

دسترسی به خطی از کد از طریق اندیس آن

در صورت استفاده از Microsoft Visual Studio، چنانچه داکیومنتی ایجاد کرده اید که خطوط فراوانی دارد و اگر می خواهید برنامه شما را مستقیم به خط معینی از کد ببرد.

Main menu را باز کرده، گزینه ی Edit -> Go To... را انتخاب کنید.

کلیدهای Ctrl + G را باهم فشار دهید.

حال، پنجره ی محاوره ای نمایان می گردد که شما می توانید با وارد کردن شماره ی خط در آن، خط مزبور را پیدا کنید.

عملگرهای اساسی C#

مقدمه

در برنامه نویسی منظور از operation، عملیاتی است که به منظور اصلاح مقدار متغیر موجود، (یا ایجاد مقداری جدید با ترکیب مقدارهای جاری)، روی مقدار معینی انجام می شود. بنابراین، operation با استفاده از حداقل یک نشانه (symbol) و یک مقدار (value) امکان پذیر می شود. نشانه ای که در عملیات به کار برده شده، operator (عملگر) خوانده می شود. حال، مقداری که در عملیات نام برده استفاده شده operand (عمل وند) گفته می شود.

Unary operator (عملگر یگانی)، عملگری است که عملیات خود را روی تنها یک عملوند انجام می دهد.چنانچه، عملگری عملیات مورد نیاز را روی دو عملوند انجام دهد، binary operand (عملگر دوگانی) اطلاق می گردد.

معرفی عملگرها و عملوندها

۱. ابتدا، برنامه ی Microsoft Visual Studio را راه اندازی کنید.

۲. به منظور ایجاد برنامه ی کاربردی جدید، به فهرست گزینه ی اصلی مراجعه کرده، سپس روی گزینه ی File -> New Project... کلیک کنید.

۳. گزینه ی Empty Project را از فهرست میانی انتخاب کنید.

۴. اسم را به gdcs4 تغییر داده، روی ok کلیک کنید.

۵. برای ایجاد فایل ویژه ی کد مورد نظر، main menu را باز کرده، سپس Project -> Add New Item... را انتخاب کنید.

۶. Code File را از لیست میانی انتخاب کنید.

۷. اسم را به CleaningOrder تغییر دهید.

۸. حال، گزینه ی Add را انتخاب کنید.

{ }

این علامت در واقع پرکاربردترین و کارامدترین عملگر در زبان C# می باشد. از این علامت به منظور ایجاد بخشی از یک کد استفاده می شود. به این ترتیب، وظیفه ی آن تعیین حد و مرز و سامان دهی فضای نامی (name space)، کلاس ها، ساختارها و استثنات است. البته، می توان آن را به دلخواه در دستورهای شرطی به کاربرد. از عملگر مزبور، به منظور ایجاد محدوده ی متغیر (variable scope) نیز استفاده می شود.

نحوه ی استفاده از { }

در فایل خالی که در اختیار شما قرار داده می شود، دستور زیر را وارد کنید.

```
class Order
{
}
```

پرانتز ( )

از عملگر پرانتز به منظور جداسازی گروهی از اقلام به کار می رود که متعلق به یک موجودیت (entity) خاص است. برای مثال، با پرانتز می توان متد Main را از یک متغیر عادی جدا کرد. مثال زیر را در نظر بگیرید.

```
class Order
{
  static void Main()
  {
  }
}
```

از عملگر پرانتز می توان به منظور جداسازی operation (عملیات) یا expression (عبارت) نیز استفاده کرد.

نحوه ی استفاده از عملگر پرانتز

برای به کاربردن این عملگر، اقدامات زیر را انجام دهید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
class Order
{
  static void Main()
  {
  }
}
```

عملگر نقطه ویرگول " ; "

علامت نقطه ویرگول، نشانگر اتمام یک عبارت یا تعریف است.

مثال

```
int number;
```

نحوه ی به کاربردن نقطه ویرگول

در فایل خالی که در اختیار شما قرار داده می شود، دستورات زیر را تایپ کنید.

```
class Order
{
  static void Main()
  {
    double mondayDiscount;
  }
}
```

عملگر ویرگول "، "

ویرگول به منظور جداسازی متغیرهای یک گروه به کار می رود. برای مثال، با کمک ویرگول می توان اسم متغیرهایی که با یک نوع داده تعریف شده را از هم جدا کرد. به مثال زیر توجه کنید.

```
class Exercise
{
  static void Main()
  {
    string firstName, lastName, fullName;
  }
}
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

از ویرگول می توان به منظور جداسازی اعضای یک enumeration یا آرگومان های یک متد نیز استفاده کرد.

استفاده از ویرگول

برای این منظور، فایل مربوطه را به صورت زیر تغییر دهید.

```
class Order
{
  static void Main()
  {
    string customerName, homePhone;
    uint numberOfShirts, numberOfPants, numberOfDresses;
    decimal priceOneShirt, priceAPairOfPants, priceOneDress;
    uint orderMonth, orderDay, orderYear;
    double mondayDiscount;
  }
}
```

عملگر جایگزین (=)

پس از تعریف متغیر، compiler مقدار مشخصی از حافظه ی کامپیوتر را به آن اختصاص می دهد. حال، آن میزان حافظه تا زمانی که شما آن را با مقدار معینی پر نکرده اید، خالی می ماند. برای قرار دادن مقداری در حافظه ی اختصاص داده شده به متغیر، می توان از عملگر جایگزین " = " بهره گرفت. دستور نحوی آن به این شکل است.

VariableName = Value

فاکتور *VariableName* باید اسم متغیر معتبری باشد. توجه داشته باشید که نمی توان از مقدار عددی (numeric value) یا رشته ای که داخل " " قرار داده شده باشد، برای این منظور استفاده کرد. در مثال زیر یک مقدار عددی به متغیر اختصاص داده.

```
class Exercise
{
  static void Main()
  {
    decimal salary;
    // Using the assignment operator
    salary = 12.55M;
  }
}
```

پس از این که متغیری تعریف شد و مقداری به آن اختصاص داده شد، می توان برای مشاهده ی (نمایش) مقدار آن Write() یا WriteLine() را فراخواند.

مثال

```
class Exercise
{
  static void Main()
  {
    decimal salary;
    // Using the assignment operator
    salary = 12.55M;
    System.Console.Write("Employee's Hourly Salary: ");
    System.Console.WriteLine(salary);
    System.Console.ReadKey();
  }
}
```

نتیجه ی زیر به دست می آید.

```
Employee's Hourly Salary: $12.55
```

کُدِ فوق ابتدا متغیر را تعریف کرده، سپس مقدار را به آن تخصیص داده است.این رویه زمانی دنبال می شود که، بخواهید مقدار یک متغیر را تغییر دهید. از فرایند تخصیص مقداری به متغیر، پس از اینکه متغیر تعریف شده، با نام مقداردهی اولیه یا initializing یاد می شود.

مثال

```
class Exercise
{
  static void Main()
  {
    // Using the assignment operator
    decimal salary = 12.55M;
    System.Console.Write("Employee's Hourly Salary: ");
    System.Console.WriteLine(salary);
    System.Console.ReadKey();
  }
}
```

همان طور که پیش تر ذکر شد، می توان چند متغیر را با استفاده از تنها یک نوع داده همزمان تعریف کرد، سپس اسامی آن ها را با ویرگول از هم جدا کرد. حین انجام این عملیات، همچنین می توان هر متغیر را با value دلخواه و پیش از عملگر ویرگول یا نقطه ویرگول، مقداردهی اولیه کرد. به مثال زیر توجه کنید.

```
class Exercise
{
  static void Main()
```

```
    {
        // Initializing various variables when declaring them with the same data type
        double value1 = 224.58, value2 = 1548.26;
        System.Console.Write("Value 1 = ");
        System.Console.WriteLine(value1);
        System.Console.Write("Value 2 = ");
        System.Console.WriteLine(value2);
        System.Console.WriteLine();
        System.Console.ReadKey();
    }
}
```

نتیجه

```
Value 1 = 224.58
Value 2 = 1548.26
```

نحوه ی تخصیص مقدار به متغیر

به منظور استفاده از عملگر جایگزین =، فایل مرتبط را این گونه اصلاح کنید.

```
class Order
{
    static void Main()
    {
        string customerName, homePhone;
        uint numberOfShirts = 5, numberOfPants = 2, numberOfDresses = 0;
        decimal priceOneShirt = 0.95M, priceAPairOfPants = 2.95M, priceOneDress = 4.55M;
        uint orderMonth = 3, orderDay = 15, orderYear = 2002;
        double mondayDiscount = 0.25; // 25%;
    }
}
```

علامت (')

از (') به منظور دخیل کردن یک کاراکتر برای مقداردهی اولیه ی متغیری به کار می رود که با نوع داده ی char تعریف شده باشد.

مثال

```
class Exercise
{
    static void Main()
    {
        char gender;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    string firstName, lastName, fullName;
    gender = 'M';
  }
}
```

نمی توان بیش از یک کاراکتر داخل ' ' قرار داد. به عبارت دیگر، جمیع نشانه های به کاربرده شده نباید به بیشتر از یک کاراکتر ارزیابی شود.

مثال

```
class Exercise
{
  static void Main()
  {
    System.Console.WriteLine('\n');
  }
}
```

علامت (")

از این علامت به منظور تعیین حد و مرز یک رشته (string) استفاده می شود. داخل " " می توان یک فضای خالی، یک کاراکتر، یک کلمه یا یک گروه کلمه قرار داد. جمیع این اقلام داخل " "، یک رشته را تشکیل می دهد. به مثال زیر توجه کنید.

```
class Exercise
{
  static void Main()
  {
    System.Console.WriteLine("The Wonderful World of C#!!!");
  }
}
```

استفاده از علامت نقل و قول

۱. برای این منظور اقدامات زیر را انجام دهید.

```
class Order
{
  static void Main()
  {
    string customerName = "James Burreck",
    homePhone = "(202) 301-7030";
    uint numberOfShirts = 1,
    numberOfPants = 1,
```

```
    numberOfDresses = 1;
    decimal priceOneShirt = 0.95M,
    priceAPairOfPants = 2.95M,
    priceOneDress = 4.55M;
    uint orderMonth = 3, orderDay = 15, orderYear = 2002;
    double mondayDiscount = 0.25; // 25%;
    System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
    System.Console.WriteLine("==========================");
    System.Console.Write("Customer: ");
    System.Console.WriteLine(customerName);
    System.Console.Write("Home Phone: ");
    System.Console.WriteLine(homePhone);
    System.Console.Write("Order Date: ");
    System.Console.Write(orderMonth);
    System.Console.Write('/');
    System.Console.Write(orderDay);
    System.Console.Write('/');
    System.Console.WriteLine(orderYear);
    System.Console.WriteLine("-------------------------");
    System.Console.WriteLine("Item Type  Qty Sub-Total");
    System.Console.WriteLine("-------------------------");
    System.Console.Write("Shirts    ");
    System.Console.Write(numberOfShirts);
    System.Console.Write("    ");
    System.Console.WriteLine(priceOneShirt);
    System.Console.Write("Pants     ");
    System.Console.Write(numberOfPants);
    System.Console.Write("    ");
    System.Console.WriteLine(priceAPairOfPants);
    System.Console.Write("Dresses   ");
    System.Console.Write(numberOfDresses);
    System.Console.Write("    ");
    System.Console.WriteLine(priceOneDress);
    System.Console.WriteLine("-------------------------");
    System.Console.Write("Monday Discount: ");
    System.Console.Write(mondayDiscount);
    System.Console.WriteLine('%');
    System.Console.WriteLine("==========================");
    System.Console.ReadKey();
  }
}
```

۲. برای مشاهده ی نتیجه برنامه را اجرا کنید. این نتیجه حاصل می گردد.

```
-/- Georgetown Cleaning Services -/-
==========================
Customer: James Burreck
Home Phone: (202) 301-7030
Order Date: 3/15/2002
------------------------
Item Type  Qty Sub-Total
------------------------
Shirts     1   0.95
Pants      1   2.95
Dresses    1   4.55
------------------------
Monday Discount: 0.25%
==========================
```

۳. حال، با زدن کلید Enter به محیط برنامه نویسی بازگردید.

کروشه []

از این علامت بیشتر زمان ها برای مدیریت و کنترل اندیس یا بُعد یک آرایه استفاده می‌شود. درباره ی کاربرد این عملگر در بخش آرایه ها بحث می کنیم.

عملگر مثبت (+)

علم جبر برای دسته بندی ارقام از یک جور خط کش استفاده می کند. وسط این خط کش عدد ۰ قرار دارد. حال، ارقامی که در سمت چپ صفر قرار می گیرند منفی تلقی می شوند و ارقامی که در سمت راست جای گرفته اند مثبت.

-∞		-6	-5	-4	-3	-2	-1		1	2	3	4	5	6		+∞
0																

-∞		-6	-5	-4	-3	-2	-1		1	2	3	4	5	6		+∞

مقداری که در سمت راست ۰ قرار دارد مثبت محسوب می‌شود. اعداد مثبت با علامت + نمایش داده می شوند : 4+ ،228+ ،90335+. در این مورد، + در واقع یک عملگر یگانی (unary operator) حساب می شود، زیرا بر تنها یک عملوند (operand) تاثیر می گذارد. توجه داشته باشید که عملگر + باید همیشه سمت چپ عملوند قرار گیرد و نه سمت راست آن.

البته، در علم ریاضی این علامت نشان داده نمی شود، و اگر عددی بدون هیچگونه علامت خاص نوشته شود، به طور معمول مثبت تلقی می گردد. پس ارقام 4+،228+ و 90335+ را می توان به این صورت نیز نوشت : 4، 228، 90335. به خاطر این که مقدار فوق هیچ گونه علامتی را به یدک نمی کشد، آن را unsigned، یا بدون علامت می خوانند.

مثال

```
class Exercise
{
  static void Main()
  {
    // Displaying an unsigned number
    System.Console.Write("Number = ");
    System.Console.WriteLine(+802);
  }
}
```

نتیجه

```
Number = 802
```

عملگر منفی (ـ)

همان طور که در خط کش بالا مشاهده می کنید، برای نشان دادن هر عددی که در سمت چپ صفر قرار می گیرد، مجبور به استفاده از علامت ـ هستیم : 12- ،448- ،32706-. عددی که با این علامت همراه باشد منفی اطلاق می شود. برخلاف عدد مثبت، عدد منفی باید با نشان مناسب (ـ) همراه باشد. برای منفی کردن یک عدد مثبت هم کافی است فقط یک علامت منفی به آن اضافه کنید.

```
class Exercise
{
  static void Main()
  {
    // Displaying an unsigned number
    System.Console.Write("First Number  ");
    System.Console.WriteLine(+802);
    // Displaying a negative number
    System.Console.Write("Second Number ");
    System.Console.WriteLine(-802);
  }
}
```

نتیجه

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

First Number 802
Second Number -802

عملگرهای یگانی : اندازه ی عملگر

زبان C# عملگر یگانی sizeof را برای محاسبه ی مقدار حافظه ی مورد نیاز یک نوع داده، در اختیار برنامه نویس قرار می دهد.

به منظور محاسبه مقدار حافظه ی مورد نیاز یک نوع داده ی خاص، باید آن (data type) را داخل پرانتز عملگر مذکور قرار داد.

مثال

```
class Exercise
{
  static void Main()
  {
    double period = 155.50;
    int size = sizeof(double);
    System.Console.Write("The value ");
    System.Console.Write(period);
    System.Console.Write(" uses ");
    System.Console.Write(size);
    System.Console.WriteLine(" bytes\n");
  }
}
```

نتیجه

The value 155.5 uses 8 bytes

زیر نمونه های بیشتری را مشاهده می کنید.

```
class Exercise
{
  static void Main()
  {
    // Thesizeof operator used to get the memory size used by
    // a variable declared with a certain a data type
    System.Console.WriteLine("The sizeof Operator");
    System.Console.WriteLine("============================");
    System.Console.WriteLine("Data Type    Memory Size");
    System.Console.WriteLine("-------------------------");
    System.Console.Write("char        ");
    System.Console.Write(sizeof(char));
    System.Console.WriteLine("  Bytes");
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
            System.Console.Write("bool       ");
            System.Console.Write(sizeof(bool));
            System.Console.WriteLine(" Bytes");
            System.Console.Write("int        ");
            System.Console.Write(sizeof(int));
            System.Console.WriteLine(" Bytes");
            System.Console.Write("uint       ");
            System.Console.Write(sizeof(uint));
            System.Console.WriteLine(" Bytes");
            System.Console.Write("short      ");
            System.Console.Write(sizeof(short));
            System.Console.WriteLine(" Bytes");
            System.Console.Write("ushort     ");
            System.Console.Write(sizeof(ushort));
            System.Console.WriteLine(" Bytes");
            System.Console.Write("byte       ");
            System.Console.Write(sizeof(byte));
            System.Console.WriteLine(" Bytes");
            System.Console.Write("sbyte      ");
            System.Console.Write(sizeof(sbyte));
            System.Console.WriteLine(" Bytes");
            System.Console.Write("float      ");
            System.Console.Write(sizeof(float));
            System.Console.WriteLine(" Bytes");
            System.Console.Write("double     ");
            System.Console.Write(sizeof(double));
            System.Console.WriteLine(" Bytes");
            System.Console.Write("decimal    ");
            System.Console.Write(sizeof(decimal));
            System.Console.WriteLine(" Bytes");
            System.Console.Write("long       ");
            System.Console.Write(sizeof(long));
            System.Console.WriteLine(" Bytes");
            System.Console.Write("ulong      ");
            System.Console.Write(sizeof(ulong));
            System.Console.WriteLine(" Bytes");
            System.Console.WriteLine("==========================");
            System.Console.WriteLine();
            System.Console.ReadKey();
        }
    }
```

نتیجه

```
The sizeof Operator
==========================
Data Type    Memory Size
--------------------------
char         2 Bytes
```

```
bool        1 Bytes
int         4 Bytes
uint        4 Bytes
short       2 Bytes
ushort      2 Bytes
byte        1 Bytes
sbyte       1 Bytes
float       4 Bytes
double      8 Bytes
decimal     16 Bytes
long        8 Bytes
ulong       8 Bytes
============================
```

عمل جمع

زمانی که بخواهیم دو چیز هم نوع را به هم اضافه کنیم (هر تعداد دفعه ای که لازم باشد)، از addition operation یا عمل جمع بهره می گیریم. گاهی نیز، به جای اضافه کردن یک چیز به چیز دیگر، یک گروه را به گروه دیگر اضافه می کنیم. تنها تفاوت آن در سرعت انجام عملیات است. در ریاضی برای انجام فرایند جمع از علامت + استفاده می شود. در C# نیز همین علامت کاربرد دارد.

برای به دست آوردن حاصل جمع دو مقدار، باید اولی را به دومی اضافه کرد. مقدار جدیدی در اختیار شما قرار می گیرد، به این صورت

value1 + value2 = value3

در مورد ارقام، نحوه و ترتیب قرارگیری اعداد چندان اهمیتی ندارد. برای مثال، value1 + value2 هیچ فرقی با value2 + value1 ندارد.

مثال

```
class Exercise
{
  static void Main()
  {
    System.Console.Write("244 + 835 = ");
    System.Console.WriteLine(244 + 835);
  }
}
```

نتیجه

```
244 + 835 = 1079
```

همچنین می توان مقدارهایی که از پیش تعریف یا مقداردهی اولیه شده را در برنامه ی خود به کار برد، حتی می توان مقدارها را از کاربر دریافت کرد.

در C#، می توان برای ایجاد یک رشته ی جدید، متغیرهای رشته را به آن اضافه کرد. عملیات همان طوری که جمع اعداد صورت می گیرد، انجام می شود. برای مثال، "Pie" + "Chart" نتیجه می دهد "PieChart" . به همین ترتیب، می توانید هر تعداد رشته که مایلید با دخیل کردن عملگر + به دستور اضافه کنید.

مثال

```
class Exercise
{
  static void Main()
  {
    varfirstName = "Alexander";
    varlastName = "Kallack";
    varfullName = firstName + " " + lastName;
    System.Console.Write("Full Name: ");
    System.Console.WriteLine(fullName);
  }
}
```

نتیجه ی زیر حاصل می گردد.

```
Full Name: Alexander Kallack
Press any key to continue...
```

استفاده از عملگر +

۱. فایل را به صورت زیر تغییر دهید.

```
class Order
  {
    static void Main()
    {
      string customerName = "James Burreck",
      homePhone = "(202) 301-7030";
      uint numberOfShirts = 1,
      numberOfPants = 1,
      numberOfDresses = 1;
      uint totalNumberOfItems;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    decimal priceOneShirt = 0.95M,
    priceAPairOfPants = 2.95M,
    priceOneDress = 4.55M;
    uint orderMonth = 3, orderDay = 15, orderYear = 2002;
    totalNumberOfItems = numberOfShirts + numberOfPants + numberOfDresses;
    System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
    System.Console.WriteLine("========================");
    System.Console.Write("Customer:  ");
    System.Console.WriteLine(customerName);
    System.Console.Write("Home Phone: ");
    System.Console.WriteLine(homePhone);
    System.Console.Write("Order Date: ");
    System.Console.Write(orderMonth);
    System.Console.Write('/');
    System.Console.Write(orderDay);
    System.Console.Write('/');
    System.Console.WriteLine(orderYear);
    System.Console.WriteLine("------------------------");
    System.Console.WriteLine("Item Type  Qty Sub-Total");
    System.Console.WriteLine("------------------------");
    System.Console.Write("Shirts     ");
    System.Console.Write(numberOfShirts);
    System.Console.Write("    ");
    System.Console.WriteLine(priceOneShirt);
    System.Console.Write("Pants      ");
    System.Console.Write(numberOfPants);
    System.Console.Write("    ");
    System.Console.WriteLine(priceAPairOfPants);
    System.Console.Write("Dresses    ");
    System.Console.Write(numberOfDresses);
    System.Console.Write("    ");
    System.Console.WriteLine(priceOneDress);
    System.Console.WriteLine("------------------------");
    System.Console.Write("Number of Items: ");
    System.Console.WriteLine(totalNumberOfItems);
    System.Console.WriteLine("========================");
    System.Console.ReadKey();
  }
}
```

۲. حال، برنامه را اجرا کنید. نتیجه ی زیر به دست می آید.

```
-/- Georgetown Cleaning Services -/-
=========================
Customer: James Burreck
Home Phone: (202) 301-7030
Order Date: 3/15/2002
------------------------
Item Type  Qty Sub-Total
------------------------
Shirts     1   0.95
Pants      1   2.95
Dresses    1   4.55
------------------------
Number of Items: 3
=========================
```

۳. پنجره ی DOS را بسته، به محیط برنامه نویسی خود بازگردید.

افزایش دادن متغییر

آسان ترین روش به منظور افزایش یک مقدار، اضافه کردن ۱ به آن است. پس از افزودن ۱ به آن، متغیر یا مقدار برای همیشه تغییر پیدا کرده، و متغیر حاوی مقدار جدید خواهد بود. مثال زیر این نمونه را به زیبایی نمایش داده است.

```
// This program studies value incrementing
  public class Exercise
  {
    static void Main()
    {
      var value = 12;
      System.Console.WriteLine("Techniques of incrementing a value");
      System.Console.Write("Value = ");
      System.Console.WriteLine(value);
      value = value + 1;
      System.Console.Write("Value = ");
      System.Console.WriteLine(value);
    }
  }
```

نتیجه

```
Techniques of incrementing a value
Value = 12
Value = 13
```

زبان C# برای این منظور، عملگر افزایش (increment operator) "++" را برای برنامه نویس فراهم می کند. به جای نوشتن Value = Value + 1، می توان نوشت Value++ و نتیجه نیز همان خواهد بود. برنامه فوق را به این ترتیب نیز می توان نوشت.

```
// This program studies value incrementing
public class Exercise
{
  static void Main()
  {
    var value = 12;
    System.Console.WriteLine("Techniques of incrementing a value");
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
    value++;
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
  }
}
```

عملگر ++ نیز یک عملگر یگانی محسوب می شود، زیرا تنها یک متغیر را دستخوش تغییر خود قرار می دهد. این عملگر با اضافه کردن 1 به متغیر، آن را تغییر می دهد. هر بار که Value++ اجرا می شود، compiler مقدار قبلی متغیر را گرفته و به آن 1 اضافه می کند. متغیر نیز سر انجام ، مقدار افزایش یافته را در خود ذخیره می کند.

```
// This program studies value incrementing
public class Exercise
{
  static void Main()
  {
    var value = 12;
    System.Console.WriteLine("Techniques of incrementing a value");
    value++;
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
    value++;
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
    value++;
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
  }
}
```

نتیجه ی زیر به دست می آید.

```
Techniques of incrementing a value
```

```
Value = 13
Value = 14
Value = 15
```

Pre and post increment

موقعیتی که عملگر در آن قرار می گیرد، به خصوص در رابطه با متغیری که اصلاح می کند، بسیار حائز اهمیت است. به منظور افزایش مقدار یک متغیر، پیش از استفاده ی دوباره از آن، باید عملگر را به طور حتم سمت چپ متغیر قرار داد.

```
// This program studies value incrementing
  public class Exercise
  {
    static void Main()
    {
      var value = 12;
      System.Console.WriteLine("Techniques of incrementing a value");
      System.Console.Write("Value = ");
      System.Console.WriteLine(value);
      System.Console.Write("Value = ");
      System.Console.WriteLine(++value);
      System.Console.Write("Value = ");
      System.Console.WriteLine(value);
    }
  }
```

نتیجه ی زیر حاصل می گردد.

```
Techniques of incrementing a value
Value = 12
Value = 13
Value = 13
```

جمع مرکب (compound addition)

افزودن یک مقدار ثابت به متغیر امری غیر عادی نیست. کافی است متغیری که قرار است میزبان مقدار جدید باشد را تعریف کنید.

مثال

```
  // This program studies value incrementing and decrementing

  public class Exercise
  {
    static void Main()
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    {
      double value = 12.75;
      double newvalue;
      System.Console.WriteLine("Techniques of incrementing and decrementing a value");
      System.Console.Write("Value = ");
      System.Console.WriteLine(value);
      newvalue = value + 2.42;
      System.Console.Write("Value = ");
      System.Console.WriteLine(newvalue);
    }
}
```

نتیجه ی زیر را به دست می دهد.

```
Techniques of incrementing and decrementing a value
Value = 12.75
Value = 15.17
```

روشی که بالا به کار رفته، ایجاب می کند شما در برنامه ی خود دو متغیر داشته باشید. مزیت آن این است که هر متغیر می تواند مقدار خود را داشته باشد، با این وجود مقدار متغیردوم وابسته است به تغییراتی که به متغییر اصلی وارد می شود. گاهی اوقات نیازی به حفظ کردن مقدار اولیه متغیر اصلی نیست، و لازم است که مقدار یک متغیر را به صورت دائمی تغییر دهید. برای این منظور، باید عملیات جمع را مستقیماً روی خود متغیر پیاده کرد (با اضافه کردن مقدار دلخواه به متغیر). با این کار نه تنها مقدار متغیر اصلاح می شود، بلکه دیگر نیازی به متغیر اضافه بر سازمان نیست.

به منظور افزودن مقدار به متغیر و اصلاح آن، می توان عملگر جایگزین را با عملگر + ترکیب کرد و عملگری جدید به وجود آورد : +=.

مثال

```
// This program studies value incrementing and decrementing
public class Exercise
{
  static void Main()
  {
    var value = 12.75;
    System.Console.WriteLine("Techniques of incrementing and decrementing a value");
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
    value += 2.42;
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
  }
}
```

برنامه همان نتیجه ی قبلی را به دست می دهد.

عملیات ضرب

ضرب به شما اجازه می دهد یک عدد را تعداد دفعات مشخص (که توسط مقدار دومی مشخص می شود) به خود آن عدد اضافه کنید. برای مثال، به جای این که مقداری را به این صورت به آن اضافه کنید : A + A + A + A، می توان ابتدا تعداد دفعاتی که این مقدار در عملیات جمع به کار رفته را پیدا کرده، سپس آن مقدار را در تعداد دفعات تکرار، ضرب کنید که در این مثال : همان عدد 4 می باشد.

درست مثل عمل جمع، ضرب نیز شرکت پذیر است : a * b * c = c * b * a. دستور نحوی عمل ضرب از همان قوانین عمل جمع پیروی می کند.

```
class Exercise
{
  static void Main()
  {
    // Initializing various variables when declaring them with the same data type
    double value1 = 224.58, value2 = 1548.26;
    var result = value1 * value2;
    System.Console.Write(value1);
    System.Console.Write(" * ");
    System.Console.Write(value2);
    System.Console.Write(" = ");
    System.Console.WriteLine(result);
    System.Console.WriteLine();
  }
}
```

نتیجه

```
224.58 * 1548.26 = 347708.2308
Press any key to continue...
```

به کاربردن عملگر ضرب

۱. برای این منظور، فایل مرتبط را به این صورت اصلاح کنید.

```
class Order
{
  static void Main()
  {
    const decimal priceOneShirt = 0.95M;
    const decimal priceAPairOfPants = 2.95M;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
const decimal priceOneDress = 4.55M;
string customerName = "James Burreck",
homePhone = "(202) 301-7030";
uint numberOfShirts = 5,
numberOfPants = 2,
numberOfDresses = 3;
uint totalNumberOfItems;
decimal subTotalShirts, subTotalPants, subTotalDresses;
decimal totalOrder;
uint orderMonth = 3, orderDay = 15, orderYear = 2002;
totalNumberOfItems = numberOfShirts + numberOfPants + numberOfDresses;
subTotalShirts = priceOneShirt * numberOfShirts;
subTotalPants = priceAPairOfPants * numberOfPants;
subTotalDresses = numberOfDresses * priceOneDress;
totalOrder = subTotalShirts + subTotalPants + subTotalDresses;
System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
System.Console.WriteLine("====================================");
System.Console.Write("Customer: ");
System.Console.WriteLine(customerName);
System.Console.Write("Home Phone: ");
System.Console.WriteLine(homePhone);
System.Console.Write("Order Date: ");
System.Console.Write(orderMonth);
System.Console.Write('/');
System.Console.Write(orderDay);
System.Console.Write('/');
System.Console.WriteLine(orderYear);
System.Console.WriteLine("------------------------------------");
System.Console.WriteLine("Item Type  Qty Unit/Price Sub-Total");
System.Console.WriteLine("------------------------------------");
System.Console.Write("Shirts    ");
System.Console.Write(numberOfShirts);
System.Console.Write("    ");
System.Console.Write(priceOneShirt);
System.Console.Write("    ");
System.Console.WriteLine(subTotalShirts);
System.Console.Write("Pants     ");
System.Console.Write(numberOfPants);
System.Console.Write("    ");
System.Console.Write(priceAPairOfPants);
System.Console.Write("    ");
System.Console.WriteLine(subTotalPants);
System.Console.Write("Dresses   ");
```

```
            System.Console.Write(numberOfDresses);
            System.Console.Write("    ");
            System.Console.Write(priceOneDress);
            System.Console.Write("    ");
            System.Console.WriteLine(subTotalDresses);
            System.Console.WriteLine("-----------------------------------");
            System.Console.Write("Number of Items: ");
            System.Console.WriteLine(totalNumberOfItems);
            System.Console.Write("Total Order:    ");
            System.Console.WriteLine(totalOrder);
            System.Console.WriteLine("===================================");
            System.Console.ReadKey();
        }
    }
```

2. برنامه را طبق دستور العمل اجرا کرده تا نتیجه ی آن را مشاهده کنید.

```
-/- Georgetown Cleaning Services -/-
===================================
Customer:  James Burreck
Home Phone: (202) 301-7030
Order Date: 3/15/2002
-----------------------------------
Item Type  Qty Unit/Price Sub-Total
-----------------------------------
Shirts     5    0.95     4.75
Pants      2    2.95     5.90
Dresses    3    4.55     13.65
-----------------------------------
Number of Items: 10
Total Order:    24.30
===================================
```

۲. اکنون، پنجره ی DOS را بسته و به محیط برنامه نویسی خود بازگردید.

ضرب مرکب

دیدیم که می توان مقداری را ازمتغیر کم کرد یا مقداری به آن اضافه کرد، سپس محصول آن را به متغیر دیگری اختصاص داد. همین عملیات را می توان برای ضرب هم پیاده کرد.

مثال

```
class Exercise
{
  static void Main()
  {
    double value = 12.75;
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
    value = value * 2.42;
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
  }
}
```

نتیجه

```
Value = 12.75
Value = 30.855
Press any key to continue...
```

به منظور آسان سازی عملیات، زبان C# عملگر ضرب مرکب (که با این علامت *= نشان داده می شود) را در اختیار برنامه نویس قرار می دهد. به مثال زیر توجه کنید.

```
class Exercise
{
  static void Main()
  {
    double value = 12.75;
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
    value *= 2.42;
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
  }
}
```

عملیات تفریق

از این عمل به منظور کم کردن مقداری از مقدار دیگر استفاده می شود. کم کردن متضاد عمل جمع محسوب می شود. عمل تفریق همان طور که می دانید با عملگر (-) انجام می شود. توجه خود را به مثال زیر جلب کنید.

```
class Exercise
{
  static void Main()
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    {
      // Values used in this program
      double value1 = 224.58, value2 = 1548.26;
      var result = value1 - value2;
      System.Console.Write(value1);
      System.Console.Write(" - ");
      System.Console.Write(value2);
      System.Console.Write(" = ");
      System.Console.WriteLine(result);
    }
  }
```

نتیجه

```
224.58 - 1548.26 = -1323.68
Press any key to continue...
```

برخلاف عمل جمع، تفریق شرکت پذیر نیست. به عبارت دیگر، a - b – c با c - b – a یکی نیست. برنامه ی زیر این حقیقت را به خوبی نشان می دهد.

```
class Exercise
{
  static void Main()
  {
    // This tests whether the addition is associative
    System.Console.WriteLine(" =+= Addition =+=");
    System.Console.Write("128 + 42 +  5 = ");
    System.Console.WriteLine(128 + 42 + 5);
    System.Console.Write("  5 + 42 + 128 = ");
    System.Console.WriteLine(5 + 42 + 128);
    System.Console.WriteLine();
    // This tests whether the subtraction is associative
    System.Console.WriteLine(" =-= Subtraction =-=");
    System.Console.Write("128 - 42 -  5 = ");
    System.Console.WriteLine(128 - 42 - 5);
    System.Console.Write("  5 - 42 - 128 = ");
    System.Console.WriteLine(5 - 42 - 128);
    System.Console.WriteLine();
  }
}
```

نتیجه ی زیر به دست می آید.

```
=+= Addition =+=
128 + 42 +  5 = 175
  5 + 42 + 128 = 175
```

```
=-= Subtraction =-=
128 - 42 -  5 = 81
 5 - 42 - 128 = -165
```

همان طور که مشاهده می کنید، هر دو عملیات یک نتیجه ی واحد و یکسان را ارائه می دهند. در بخش منها عملیات، اعداد همان ترتیب را دنبال می کنند، ولی نتایج متفاوتی به دست می دهند.

به کاربردن عملگر منها

1. برای انجام عمل منها، فایل را به صورت زیر تغیر دهید.

```
class Order
 {
   static void Main()
   {
     const decimal priceOneShirt = 0.95M;
     const decimal priceAPairOfPants = 2.95M;
     const decimal priceOneDress = 4.55M;
     const decimal salestaxRate = 0.0575M; // 5.75%
     string customerName = "James Burreck",
     homePhone = "(202) 301-7030";
     uint numberOfShirts = 5,
     numberOfPants = 2,
     numberOfDresses = 3;
     uint totalNumberOfItems;
     decimal subTotalShirts, subTotalPants, subTotalDresses;
     decimal taxAmount, totalOrder, netPrice;
     uint orderMonth = 3, orderDay = 15, orderYear = 2002;
     totalNumberOfItems = numberOfShirts + numberOfPants + numberOfDresses;
     subTotalShirts = priceOneShirt * numberOfShirts;
     subTotalPants = priceAPairOfPants * numberOfPants;
     subTotalDresses = numberOfDresses * priceOneDress;
     totalOrder = subTotalShirts + subTotalPants + subTotalDresses;
     taxAmount = totalOrder * salestaxRate;
     netPrice = totalOrder - taxAmount;
     System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
     System.Console.WriteLine("=====================================");
     System.Console.Write("Customer:  ");
     System.Console.WriteLine(customerName);
     System.Console.Write("Home Phone: ");
     System.Console.WriteLine(homePhone);
     System.Console.Write("Order Date: ");
     System.Console.Write(orderMonth);
     System.Console.Write('/');
```

```
        System.Console.Write(orderDay);
        System.Console.Write('/');
        System.Console.WriteLine(orderYear);
        System.Console.WriteLine("-------------------------------------");
        System.Console.WriteLine("Item Type  Qty Unit/Price Sub-Total");
        System.Console.WriteLine("-------------------------------------");
        System.Console.Write("Shirts    ");
        System.Console.Write(numberOfShirts);
        System.Console.Write("    ");
        System.Console.Write(priceOneShirt);
        System.Console.Write("    ");
        System.Console.WriteLine(subTotalShirts);
        System.Console.Write("Pants     ");
        System.Console.Write(numberOfPants);
        System.Console.Write("    ");
        System.Console.Write(priceAPairOfPants);
        System.Console.Write("    ");
        System.Console.WriteLine(subTotalPants);
        System.Console.Write("Dresses   ");
        System.Console.Write(numberOfDresses);
        System.Console.Write("    ");
        System.Console.Write(priceOneDress);
        System.Console.Write("    ");
        System.Console.WriteLine(subTotalDresses);
        System.Console.WriteLine("-------------------------------------");
        System.Console.Write("Number of Items: ");
        System.Console.WriteLine(totalNumberOfItems);
        System.Console.Write("Total Order:   ");
        System.Console.WriteLine(totalOrder);
        System.Console.Write("Tax Rate:     ");
        System.Console.Write(salestaxRate * 100);
        System.Console.WriteLine('%');
        System.Console.Write("Tax Amount:    ");
        System.Console.WriteLine(taxAmount);
        System.Console.Write("Net Price:    ");
        System.Console.WriteLine(netPrice);
        System.Console.WriteLine("=====================================");
        System.Console.ReadKey();
    }
}
```

2. برنامه را اجرا کنید. نتیجه ی زیر به دست می آید.

```
-/- Georgetown Cleaning Services -/-
=====================================
Customer: James Burreck
Home Phone: (202) 301-7030
```

```
Order Date: 3/15/2002
-------------------------------------
Item Type  Qty Unit/Price Sub-Total
-------------------------------------
Shirts     5   0.95    4.75
Pants      2   2.95    5.90
Dresses    3   4.55    13.65
-------------------------------------
Number of Items: 10
Total Order:    24.30
Tax Rate:       5.7500%
Tax Amount:     1.397250
Net Price:      22.902750
=====================================
```

3. پنجره ی DOS را بسته و به محیط برنامه نویسی بازگردید.

کاهش دادن متغیر

زمانی که اعداد را به صورت معکوس می شمارید، مثل 8 ،7 ،6 ،5 ، در واقع یک 1 رقم از آن کسر می کنید. از این عملیات با نام کاهش مقدار (decrementing a value) یاد می شود.

```
// This program studies value decrementing
public class Exercise
{
  static void Main()
  {
    var value = 12;
    System.Console.WriteLine("Techniques of decrementing a value");
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
    value = value - 1;
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
  }
}
```

نتیجه

```
Techniques of decrementing a value
Value = 12
Value = 11
```

زبان برنامه نویسی C#، برای این منظور عملگر کاهش (--) را ارائه می دهد. با استفاده از عملگر مذکور، عملیات فوق را می توان به این صورت اصلاح کرد.

```
public class Exercise
{
  static void Main()
  {
    var value = 12;
    System.Console.WriteLine("Techniques of decrementing a value");
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
    value--;
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
  }
}
```

Pre-decrementing a variable

در این مورد نیز موقعیت عملگر (جایی که در آن قرار می گیرد) اهمیت به سزایی دارد. چنانچه مایلید متغیر را پیش از فراخوانی آن کاهش دهید، باید عملگر کاهش (decrement operator) را سمت چپ عملوند قرار دهید. برنامه ی زیر این عملیات را به تصویر کشیده.

```
// This program studies value decrementing
public class Exercise
{
  static void Main()
  {
    var value = 12;
    System.Console.WriteLine("Techniques of decrementing a value");
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
    System.Console.Write("Value = ");
    System.Console.WriteLine(--value);
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
  }
}
```

نتیجه

```
Techniques of decrementing a value
Value = 12
Value = 12
```

```
Value = 11
```

تفریق مرکب (compound subtraction)

به منظور کسر مقدار ثابتی (constant value) از یک متغیر، از عملگر -= استفاده می شود.

مثال

```
// This program studies value incrementing and decrementing
public class Exercise
{
  static void Main()
  {
    var value = 12.75;
    System.Console.WriteLine("Techniques of incrementing and decrementing a value");
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
    value -= 2.42;
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
  }
}
```

نتیجه

```
Techniques of incrementing and decrementing a value
Value = 12.75
Value = 10.33
```

عملیات تقسیم (division operation)

تقسیم یک مقدار عبارتند از جداسازی آن به بخش های متعدد. برای مثال، هنگامی که سیبی را از وسط نصف می کنید، در واقع آن را به دو بخش تقسیم کرده اید. عمل تقسیم با عملگر (/) انجام می گیرد.

مثال

```
class Exercise
{
  static void Main()
  {
    // Initializing various variables when declaring them with the same data type
```

```
    double value1 = 224.58, value2 = 1548.26;
    var result = value1 / value2;
    System.Console.Write(value1);
    System.Console.Write(" / ");
    System.Console.Write(value2);
    System.Console.Write(" = ");
    System.Console.WriteLine(result);
    System.Console.WriteLine();
  }
}
```

این مثال نتیجه ی زیر را ارائه می دهد.

```
224.58 / 1548.26 = 0.145053156446592
Press any key to continue...
```

نکته: هیچ گاه چیزی را به صفر تقسیم نکنید.

به کاربر بردن عملگر تقسیم

۱. برای این منظور، فایل را به این صورت تغییر دهید.

```
class Order
  {
    static void Main()
    {
      const decimal priceOneShirt = 0.95M;
      const decimal priceAPairOfPants = 2.95M;
      const decimal priceOneDress = 4.55M;
      const decimal discountRate = 0.20M; // 20%
      const decimal taxRate = 5.75M;  // 5.75%
      string customerName = "James Burreck", homePhone = "(202) 301-7030";
      uint numberOfShirts = 5, numberOfPants = 2, numberOfDresses = 3;
      uint totalNumberOfItems;
      decimal subTotalShirts, subTotalPants, subTotalDresses;
      decimal DiscountAmount, totalOrder, netPrice, taxAmount, salesTotal;
      decimal amountTended, difference;
      uint orderMonth = 3, orderDay = 15, orderYear = 2002;
      totalNumberOfItems = numberOfShirts + numberOfPants + numberOfDresses;
      subTotalShirts = priceOneShirt * numberOfShirts;
      subTotalPants = priceAPairOfPants * numberOfPants;
      subTotalDresses = numberOfDresses * priceOneDress;
      totalOrder = subTotalShirts + subTotalPants + subTotalDresses;
```

```
DiscountAmount = totalOrder * discountRate;
netPrice = totalOrder - DiscountAmount;
taxAmount = totalOrder * taxRate / 100;
salesTotal = netPrice + taxAmount;
amountTended = 50M;
difference = amountTended - salesTotal;
System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
System.Console.WriteLine("==================================");
System.Console.Write("Customer: ");
System.Console.WriteLine(customerName);
System.Console.Write("Home Phone: ");
System.Console.WriteLine(homePhone);
System.Console.Write("Order Date: ");
System.Console.Write(orderMonth);
System.Console.Write('/');
System.Console.Write(orderDay);
System.Console.Write('/');
System.Console.WriteLine(orderYear);
System.Console.WriteLine("-----------------------------------");
System.Console.WriteLine("Item Type  Qty Unit/Price Sub-Total");
System.Console.WriteLine("-----------------------------------");
System.Console.Write("Shirts    ");
System.Console.Write(numberOfShirts);
System.Console.Write("    ");
System.Console.Write(priceOneShirt);
System.Console.Write("    ");
System.Console.WriteLine(subTotalShirts);
System.Console.Write("Pants     ");
System.Console.Write(numberOfPants);
System.Console.Write("    ");
System.Console.Write(priceAPairOfPants);
System.Console.Write("    ");
System.Console.WriteLine(subTotalPants);
System.Console.Write("Dresses   ");
System.Console.Write(numberOfDresses);
System.Console.Write("    ");
System.Console.Write(priceOneDress);
System.Console.Write("    ");
System.Console.WriteLine(subTotalDresses);
System.Console.WriteLine("-----------------------------------");
System.Console.Write("Number of Items: ");
System.Console.WriteLine(totalNumberOfItems);
System.Console.Write("Total Order:   ");
```

```
        System.Console.WriteLine(totalOrder);
        System.Console.Write("Discount Rate: ");
        System.Console.Write(discountRate * 100);
        System.Console.WriteLine('%');
        System.Console.Write("Discount Amount: ");
        System.Console.WriteLine(DiscountAmount);
        System.Console.Write("After Discount: ");
        System.Console.WriteLine(netPrice);
        System.Console.Write("Tax Rate:    ");
        System.Console.Write(taxRate);
        System.Console.WriteLine('%');
        System.Console.Write("Tax Amount:    ");
        System.Console.WriteLine(taxAmount);
        System.Console.Write("Net Price:    ");
        System.Console.WriteLine(salesTotal);
        System.Console.WriteLine("==================================");
        System.Console.Write("Amount Tended: ");
        System.Console.WriteLine(amountTended);
        System.Console.Write("Difference:    ");
        System.Console.WriteLine(difference);
        System.Console.WriteLine("==================================");
        System.Console.ReadKey();
    }
}
```

۲. برنامه را اجرا کنید. نتیجه ی زیر را به دست می دهد.

```
-/- Georgetown Cleaning Services -/-
==================================
Customer: James Burreck
Home Phone: (202) 301-7030
Order Date: 3/15/2002
----------------------------------
Item Type  Qty Unit/Price Sub-Total
----------------------------------
Shirts    5   0.95    4.75
Pants     2   2.95    5.90
Dresses   3   4.55    13.65
----------------------------------
Number of Items: 10
Total Order:   24.30
Discount Rate:  20.00%
Discount Amount: 4.8600
```

```
After Discount：19.4400
Tax Rate：    5.75%
Tax Amount：  1.39725
Net Price：   20.83725
==================================
Amount Tendered：50
Difference：  29.16275
==================================
```

۳. پنجره ی DOS را بسته و به محیط برنامه نویسی برگردید.

تقسیم مرکب (Compound division)

به خاطر دارید که می توان مقداری را به متغیری اضافه کرد، در آن ضرب کرد، یا از آن کسر کرد، سپس نتیجه ی آن را به خود متغیر اختصاص داد. این عملیات را می توان برای تقسیم نیز پیاده کرد. نمونه ی آن را در زیر مشاهده می کنید.

```
class Exercise
{
  static void Main()
  {
    double value = 12.75;
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
    value = value / 2.42;
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
  }
}
```

نتیجه ی زیر حاصل می گردد.

```
Value = 12.75
Value = 5.26859504132231
Press any key to continue...
```

زبان C# برای این منظور عملگر /= را ارائه می دهد. مثال زیر این عملگر را به کار می برد.

```
class Exercise
{
  static void Main()
  {
```

```
    double value = 12.75;
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
    value /= 2.42;
    System.Console.Write("Value = ");
    System.Console.WriteLine(value);
  }
}
```

باقی مانده (Remainder)

عملیات تقسیم نتیجه ای با رقم اعشار در اختیار شما قرار می دهد، چنانچه عددی فرد (مثل ۱۴۷) در آن به کار ببرید. گاهی اوقات به باقی مانده ی عمل تقسیم نیاز داریم. برای مثال، تصور کنید ۲۶ بازیکن کودک در استادیوم حضور دارند، و بازی در شرف آغاز است. مطلع هستید که برای هر تیم به ۱۱ بازیکن نیاز است. اگر بازی با تعداد صحیح شروع شود، چند نفر بیرون از بازی منتظر می مانند؟

عملگر باقی مانده (remainder operation) که با علامت (%) نشان داده می شود، این عملیات را انجام می دهد. عملگر فوق با گرفتن Shift + 5، فعال می شود.

مثال

```
class Exercise
{
  static void Main()
  {
    var players = 18;
    var remainder = players % 11;
    // When the game starts, how many players will wait?.
    System.Console.Write("Out of ");
    System.Console.Write(players);
    System.Console.Write(" players, ");
    System.Console.Write(remainder);
    System.Console.WriteLine(" players will have to wait when the game starts.\n");
  }
}
```

نتیجه ی زیر به دست می آید.

```
Out of 18 players, 7 players will have to wait when the game starts.
Press any key to continue...
```

باقی مانده ی مرکب (compound remainder)

همان طور که در عملگرهای دیگر ریاضی مشاهده کردید، می توان پس از به دست آوردن باقی مانده ی یک متغیر نتیجه ی آن را دوباره به خود متغیر تخصیص داد.

مثال

```
class Exercise
{
  static void Main()
  {
    var players = 18;
    // When the game starts، how many players will wait?.
    System.Console.Write("Out of ");
    System.Console.Write(players);
    System.Console.Write(" players، ");
    players = players % 11;
    System.Console.Write(players);
    System.Console.WriteLine(" players will have to wait when the game starts.\n");
  }
}
```

زبان C# با فراهم کردن عملگر %=، فرایند را سرعت می بخشد.

مثال

```
class Exercise
{
  static void Main()
  {
    var players = 18;
    // When the game starts، how many players will wait?.
    System.Console.Write("Out of ");
    System.Console.Write(players);
    System.Console.Write(" players، ");
    players %= 11;
    System.Console.Write(players);
    System.Console.WriteLine(" players will have to wait when the game starts.\n");
  }
}
```

Bit Operations

همان طور که در دروس پیشین ذکر کردیم، کامپیوتر اطلاعات دریافتی را در قسمت های بسیار کوچکی به نام Bit در حافظه ی خود ذخیره می کند. هر Bit در خود یا ۰ دارد یا ۱. می توان Bit را با جا به جایی مقدار های ۰ و۱ (۰ به ۱ یا ۱ به ۰) دستکاری کرد.

عملیاتی که روی Bit انجام می گیرد، فقط با ۰ و ۱ سروکار دارند. به عبارت دیگر، هر عدد مبنای ۱۶ (hexadecimal) یا دهدهی (decimal) باید ابتدا به دودویی (binary) تبدیل شود.

عملیاتی که در برنامه نویسی Microsoft Windows (Win32) کاربرد فراوان دارد، OR operation است که به آن خواهیم پرداخت.

"معکوس کردن" Bit (Reversing " a bit ")

به خاطر دارید که هر Bit، تنها یک مقدار دربردارد : یا ۰ و یا ۱. وارونه سازی (معکوس کردن) جزئی از عملیات ایجاد تغییر در Bit می باشد. برای مثال، اگر در یک Bit مقدار ۱ را دارید، به ۰ تبدیل می شود و اگر مقدار ۰ را دارید به ۱ تبدیل می شود. زبان C# با عملگر ~، این عملیات را ممکن می سازد.

برای نمونه رقم ۲۸۶ را در نظر بگیرید. این رقم (decimal دهدهی است) با سیستم دودویی (binary) به این شکل در می آید : 100011110. حال، می توان هر bit را به صورت زیر وارونه کرد.

286	1	0	0	0	1	1	1	1	0
Not 286	0	1	1	1	0	0	0	0	1

برای استفاده ی صحیح از عملگر not، ~ را در سمت چپ مقدار قرار دهید.

مثال

```
class Exercise
{
  static void Main()
  {
    var number1 = 286;
    System.Console.Write("286 = ");
    System.Console.WriteLine(number1);
    System.Console.Write("Not 286 = ");
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
        System.Console.WriteLine(~number1);
    }
}
```

نتیجه ای که حاصل می شود.

```
286 = 286
Not 286 = -287
Press any key to continue...
```

پیوستگی بیتی (Bitwise Conjunction)

عبارت است از پیوند دادن (یا ضمیمه کردن) محتوای یک bit به bit دیگر. برای این منظور، C# عملگر & را تعبیه کرده.

برای پیوند دادن محتوای دو bit، به خاطر داشته باشید که اول باید آن ها را به سیستم دودویی (binary) تبدیل کنید.

چنانچه bit ای با مقدار ۰ به bit دیگری با همین مقدار اضافه شود، نتیجه ۰ خواهد بود.

Bit0	0
Bit1	0
Bit0 And Bit1	0

چنانچه bit با مقدار ۱، به bit ای با مقدار ۰ اضافه شود، بازهم نتیجه ۰ خواهد بود.

Bit0	1
Bit1	0
Bit0 And Bit1	0

اگر bit با مقدار ۰، به bit ای با مقدار ۱ اضافه شود، نتیجه ۰ خواهد بود.

Bit0	0
Bit1	1

Bit0 And Bit1	0

چنانچه bit ای با مقدار ۱ به bit دیگری با همین مقدار اضافه شود، نتیجه ۱ خواهد شد.

Bit0	1
Bit1	1
Bit0 And Bit1	1

تصور کنید می خواهید عدد 286 bit را به ۴۷۵ اضافه کنید. رقم دهدهی ۲۸۶ را که به سیستم دودویی تبدیل کنید، این نتیجه به دست می آید : 100011110 .نسخه ی دودویی رقم 4075 (که دهدهی می باشد) معادل : 111111101011 می باشد. بر اساس چهار قاعده ی فوق، می توان این دو رقم را به صورت زیر جمع کرد.

286	0	0	0	1	0	0	0	1	1	1	1	0
4075	1	1	1	1	1	1	1	0	1	0	1	1
286 & 4075	0	0	0	1	0	0	0	0	1	0	1	0

بنابراین، 4075 & 286 نتیجه ی 100001010 را می دهد، برابر است با

	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	256	128	64	32	16	8	4	2	1
286 & 4075	1	0	0	0	0	1	0	1	0
	256	0	0	0	0	8	0	2	0

یعنی ؛ 286 & 4075 = 256 + 16 + 2 = 266. حال به منظور اجرای عملیات پوستگی بیتی، از عملگر & کمک می گیریم.

class Exercise

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
{
  static void Main()
  {
    var number1 = 286;
    var number2 = 4075;
    var result = number1 & number2;
    System.Console.Write("286 & 4075 = ");
    System.Console.WriteLine(result);
  }
}
```

نتیجه

```
286 & 4075 = 266
Press any key to continue...
```

همچنین می توان پس از انجام عملیات بر روی متغیر (پیوست دادن دو bit)، نتیجه را دوباره برای همان متغیر به کار برد. به مثال زیر توجه کنید.

```
class Exercise
{
  static void Main()
  {
    var number = 286;
    System.Console.WriteLine(number);
    number = number & 48;
    System.Console.WriteLine(number);
  }
}
```

نتیجه

```
286
16
Press any key to continue...
```

می توان عملیات فوق را با به کاربردن عملگر =&، به تنها یک مرحله خلاصه کرد.

```
class Exercise
{
  static void Main()
  {
    var number = 286;
    System.Console.WriteLine(number);
    number &= 48;
    System.Console.WriteLine(number);
```

```
    }
}
```

جداسازی بیتی (Bitwise Disjunction)

عبارت است از جدا کردن یک bit از bit ای دیگر. C# با فراهم کردن عملگر (|)، این امکان را فراهم می سازد.

همان طور که به خاطر دارید برای اضافه کردن دو رقم (از طریق پیوند دادن دو bit)، باید ابتدا آن ها را به سیستم دودویی تبدیل کرد. سپس

اگر bit ای با مقدار ۰ به bit دیگری با همان مقدار اضافه شود، نتیجه ۰ می شود.

Bit0	0
Bit1	0
Bit0 Or Bit1	0

چنانچه bit ای با مقدار ۱ به bit دیگری با مقدار ۰ اضافه شود، نتیجه ۱ می شود.

Bit0	1
Bit1	0
Bit0 Or Bit1	1

اگر bit ای با مقدار ۰ به bit دیگری با مقدار ۱ اضافه شود، نتیجه ۱ می شود.

Bit0	0
Bit1	1
Bit0 Or Bit1	1

حال، چنانچه بیتی با مقدار ۱ به bit دیگری با همین مقدار اضافه شود، نتیجه ۱ خواهد شد.

Bit0	1
Bit1	1

Bit0 Or Bit1	1

به عنوان مثال، رقم ۳۰۵ را می خواهیم از 2853 جدا کنیم. نسخه ی دودویی 305 (رقمی دهدهی) برابر است با 100110001. رقم دهدهی 2853، در سیستم دودیی به این صورت خواهد بود101100100101 . بر مبنای چهار قاعده ی فوق، می توان این دو رقم را به صورت زیر جداسازی کرد.

305	0	0	0	1	0	0	1	1	0	0	0	1
2853	1	0	1	1	0	0	1	0	0	1	0	1
305 | 2853	1	0	1	1	0	0	1	1	0	1	0	1

بنابراین، 2853 | 305 نتیجه می دهد : 101100110101 که در آخر برابر است با

	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	2048	1024	512	256	128	64	32	16	8	4	2	1
305 Or 2853	1	0	1	1	0	0	1	1	0	1	0	1
	2048	0	512	256	0	0	32	16	0	4	0	1

یعنی ؛ 2869 = 1 + 4 + 16 + 32 + 256 + 512 + 2048 = 4075 | 286.

عملیات فوق را می توان به صورت زیر نیز محاسبه کرد.

```
class Exercise
  {
    static void Main()
    {
      var number1 = 305;
      var number2 = 2853;
      var result = number1 | number2;
      System.Console.Write("305 Or 2853 = ");
      System.Console.WriteLine(result);
    }
  }
```

نتیجه

```
305 Or 2853 = 2869
Press any key to continue…
```

می توان تعدادی از bit ها را در یک متغیر جداسازی کرد، سپس نتیجه را به همان متغیر اختصاص داد.

```
class Exercise
{
  static void Main()
  {
    var number = 305;
    System.Console.WriteLine(number);
    number = number | 22;
    System.Console.WriteLine(number);
  }
}
```

نتیجه

```
305
311
Press any key to continue...
```

نسخه ی فشرده ی این عملیات با به کارگیری عملگر =|، امکان پذیر می باشد.

```
class Exercise
{
  static void Main()
  {
    var number = 305;
    System.Console.WriteLine(number);
    number |= 22;
    System.Console.WriteLine(number);
  }
}
```

Bitwise Exclusion

فرایندی است که در آن دو bit با در نظر گرفتن قوانین زیر به هم اضافه می شوند (زبان C# برای این منظور عملگر ^ را ارائه می دهد).

چنانچه هر دو bit دارای مقدار یکسانی باشند، نتیجه ۰ خواهد بود.

Bit0	0	1

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

Bit1	0	1
Bit0 ^ Bit1	0	0

اگر دو bit مقدار متفاوتی دارند، نتیجه ۱ می شود.

Bit0	0	1
Bit1	1	0
Bit0 ^ Bit1	1	1

فرض کنید می خواهیم مقدار 618 را از 2548 خارج کنیم. معادل دودویی 618 : 1001101010 . معادل دودویی 2548 : 100111110100 .

بر مبنای دو قاعده ی فوق، می توان bit های رقم 618 را از 2548 خارج کرد.

618	0	0	1	0	0	1	1	0	1	0	1	0
2548	1	0	0	1	1	1	1	1	0	1	0	0
618 ^ 2548	1	0	1	1	1	0	0	1	1	1	1	0

بنابراین، 2853 ^ 305 نتیجه می دهد : 101110011110 که برابر است با

	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	2048	1024	512	256	128	64	32	16	8	4	2	1
618 ^ 2548	1	0	1	1	1	0	0	1	1	1	1	0
	2048	0	512	256	128	0	0	16	8	4	2	0

یعنی : 2974 = 2 + 4 + 8 + 16 + 128 + 256 + 512 + 2048 = 4075 ^ 286 .

البته، عمل فوق را به این صورت نیز می توان محاسبه کرد.

```
class Exercise
{
  static void Main()
  {
    var number1 = 618;
    var number2 = 2548;
    var result = number1 ^ number2;
    System.Console.WriteLine(result);
  }
}
```

نتیجه

```
2974
Press any key to continue...
```

می توان شماری bit را به صورتی که گفته شد، از متغیری خارج کرد و بعد نتیجه ی حاصل را برای خود متغیر به کاربرد.

مثال

```
class Exercise
{
  static void Main()
  {
    var number = 618;
    System.Console.WriteLine(number);
    number = number ^ 38;
    System.Console.WriteLine(number);
  }
}
```

نتیجه می دهد.

```
618
588
Press any key to continue...
```

همچنین می توان با استفاده از عملگر =^ به همین نتیجه دست یافت.

```
class Exercise
  {
    static void Main()
    {
      var number = 618;
      System.Console.WriteLine(number);
```

```
        number ^= 38;
        System.Console.WriteLine(number);
    }
}
```

انتقال Bit ها از راست به چپ

عبارت است از جا به جایی bit ها از سمت راست به سمت چپ. در این مورد نیز، لازم است پیش از انجام عملیات عدد به معادل دودویی خود تبدیل شود. C# این عملیات را با عملگر >> پشتیبانی می کند.

رقم ۷۴۱ را در نظر بگیرید. معادل دودویی آن برابر است با : 1011100101 که به صورت زیر نمایش داده می شود.

1	0	1	1	1	0	0	1	0	1

به منظور انجام عملیات فوق، باید تک تک مقدارها را (بسته به تعداد دفعات جا به جایی) از سمت راست به چپ جا به جا کرد. با انتقال یک bit از چپ به راست، موقعیت تنها یک bit تغییر می کند. برای مثال، bit ای در موقعیت x قرار می گیرد و bit دیگری سمت چپ آن در موقعیت y. مقدار bit x را جایگزین مقدار bit y می کنیم. در صورتی که bit x در راست ترین موقعیت قرار داشت، مقدار ۰ دریافت می کند.

Original		1	0	1	1	1	0	0	1	0	1
<< by 1	1	0	1	1	1	0	0	1	0	1	0

نتیجه ی 10111001010 را به دست می دهد. نتیجه ی دهدهی (decimal) آن به این صورت محاسبه شده.

	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	1024	512	256	128	64	32	16	8	4	2	1
741 << 1	1	0	1	1	1	0	0	1	0	1	0
	1024	0	256	128	64	0	0	8	0	2	0

همچنین، این نتیجه حاصل می شود.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

741 << 1 = 1024 + 256 + 128 + 64 + 8 + 2 = 1482

همان طور که از مثال بالا پیدا است، باید برای پیاده سازی این عملیات از عملگر << استفاده کرد.

مثال

```
class Exercise
{
  static void Main()
  {
    System.Console.WriteLine(741 << 1);
  }
}
```

محصول فرایند

```
1482
Press any key to continue...
```

به همین ترتیب، می توان bit ها را بیش از یک واحد به سمت چپ انتقال داد.

```
class Exercise
{
  static void Main()
  {
    var number = 248 << 5;
    System.Console.WriteLine(number);
  }
}
```

نتیجه

```
7936
Press any key to continue...
```

بنابراین، می توان چند bit را به سمت چپ (یک متغیر) جا به جا کرد، سپس نتیجه ی حاصله را برای خود متغیر اعمال کرد.

```
class Exercise
{
  static void Main()
  {
    var number = 248;
    System.Console.WriteLine(number);
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    number = number << 5;
    System.Console.WriteLine(number);
  }
}
```

نتیجه ی زیر را به دست می دهد.

```
248
7936
Press any key to continue...
```

C# عملیات فوق را با به کاربردن عملگر =<< برای کاربر بسیار آسان می کند.

```
class Exercise
{
  static void Main()
  {
    var number = 248;
    System.Console.WriteLine(number);
    number <<= 5;
    System.Console.WriteLine(number);
  }
}
```

انتقال Bit ها به سمت راست

می توان bit ها را در سمت راست قرار داد. برای این منظور تمام مراحلی که برای انتقال bit ها به چپ انجام می دادید را به صورت معکوس برای این فرایند پیاده کنید. زبان C# با عملگر >>، این پروسه را امکان پذیر می کند.

کلاس ها

برای تعریف متغیر، می توان از کلیدواژه ی var یا نوع داده ی ساده و شناخته شده بهره گرفت. برای مثال، می توان برای تعریف متغیری که نشانگر تعداد اتاق خواب های یک خانه است، از integer استفاده کرد.

```
class Program
{
  static void Main()
  {
    int bedrooms = 3;
  }
}
```

در C# می توان با به کاربردن چند متغیر، شی ای کامل پیچیده ساخت. در زبان های برنامه نویسی کلاس (که خود متشکل از یک گروه متغیر است)، در واقع پایه ای برای ایجاد متغیری پیچیده تر می باشد. حال برای به وجود آوردن یک کلاس، ابتدا کلیدواژه ی class را تایپ کرده،سپس به دنبال آن اسم و بدنه ی اصلی را داخل {} قرار داد.

نحوه ی وارد کردن کلاس

۱. Microsoft Visual Studio را اجرا کنید.
۲. به main menu مراجعه کرده و برای ایجاد app جدید گزینه های File -> New Project را انتخاب کنید.
۳. در فهرست میانی روی گزینه ی Empty Project کلیک کنید.
۴. اسم مورد نظر را به DepartmentStore1 تغییر دهید.
۵. حال OK را کلیک کنید.

نام گذاری کلاس

درست مثل متغیر، یک کلاس باید دارای اسم باشد. اسم گذاری برای کلاس بسیار شبیه به فرایند اسم گذاری برای متغیر است (از همان قوانین پیروی می کند). البته، شما می توانید قوانین خو را به کار ببرید. در دروس ما

چنانچه اسم کلاس از تنها یک کلمه تشکیل شده، فقط حرف اول با حروف بزرگ نوشته می شود. نمونه های آن :
Identification، Vehicle، Student می باشد.

اگر اسم کلاس موردنظر از چند کلمه ساخته شده باشد، تمام حروف اول کلمات با حرف بزرگ نوشته می شوند : DrivingRecord، SocialSecurityInformation، GeometricShape

مدیریت کلاس ها

Microsoft Visual Studio ابزار و امکان های بصری زیادی برای مدیریت کلاس در اختیار برنامه نویس می گذارد. همچنین فرایند نام گذاری یا تغییر اسم کلاس ها را با امکاناتی که دارد برای کاربر آسان ساخته است.

پنجره ی Class View

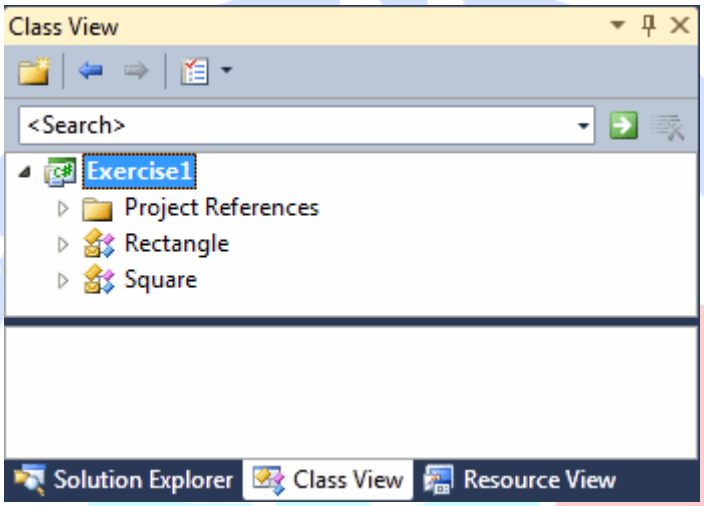
بهترین روش برای مدیریت کلاس، استفاده از پنجره ی Class View است. برای دسترسی به این پنجره

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

چنانچه پنجره از قبل باز نیست، ابتدا main menu را باز کرده، سپس View -> Class View را کلیک کنید.

اگر پنجره ی مذکور از قبل باز بود، تنها کافی است روی tab آن کلیک کنید.

پنجره ی Class View از شش بخش مجزا تشکیل شده. همانطور که در تصویر زیر مشاهده می کنید کاربرد و نوار عنوان آن مشابه solution explorer می باشد.



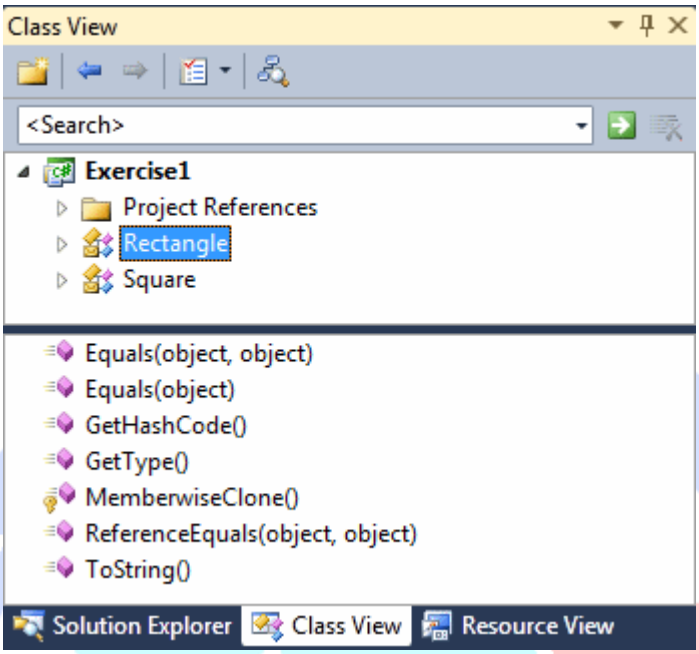
دومین بخش تشکیل دهنده ی نوار عنوان، نوار ابزار (toolbar) آن می باشد  .

همان طور که از اسم آن پیدا است گزینه ی New Folder به شما اجازه می دهد فولدری جدید ایجاد کرده و به پروژه اضافه کنید. دکمه های Back و Forward به شما اجازه می دهند در کلاس ها پیمایش کنید.

در زیر toolbar، نوار دیگری وجود دارد که از یک combo box و دکمه تشکیل شده که به کمک آن ها می توانید جستجو کنید.

بدنه ی اصلی پنجره ی Class View نیز از دو بخش اصلی تشکیل شده. اولین گره ی قسمت بالایی پنجره، اسم پروژه را نشان می دهد. حال زیر گره ی پروژه، اسم کلاس ها نمایش داده شده است. قسمت پایینی پنجره نیز به اعضای کلاس ها اختصاص داده شده است.

برای مشاهده ی اعضای کلاس ها



ایجاد کلاس

راه های مختلفی برای ساختن کلاس وجود دارد. در صورت کار با برنامه های ویرایش متن مثل Notepad، فایل را ایجاد کرده، کلیدواژه ی class را تایپ کنید، به دنبال آن اسم کلاس مورد نظر و علامت { را درج کنید.

نمونه

```
class House
{
}
```

حال، فایل خود را با پسوند .cs ذخیره کنید. البته، نیازی نیست که اسم فایل با اسم کلاس یکی باشد. به خاطر داشته باشید که باید فایل مزبور را در همان فولدری که فایل های دیگر پروژه در آن نگه داری می شوند، ذخیره کنید.

کلاس در یک code file ایجاد می شود. به همین خاطر هم می توانید آن را در اولین فایل پروژه ی خود ذخیره کنید.

```
class House
{
}
class Exercise
{
  static void Main()
  {
    int bedrooms = 3;
```

```
    }
}
```

Microsoft Visual C# 2010 Express یا Microsoft Visual Studio فرایند ایجاد کلاس را برای شما بسیار آسان می سازد. در درس ۱، دیدیم که چگونه می توان code file ایجاد کرد. پس از ساختن code file نام برده، کافی است کد مورد نیاز را برای کلاس وارد کنید. پس از انتخاب گزینه ی Add New Item (در solution explorer یا main menu)، می توان نوع فایل دلخواه را انتخاب کرد از جمله فایل کلاس.

به منظور ایجاد کلاسی جدید

در main menu روی گزینه ی Project -> Add Class... کلیک کنید، (یا در لیست میانی پنجره ی محاوره Add New Item، Project -> Add New Item، روی گزینه ی class کلیک کنید).

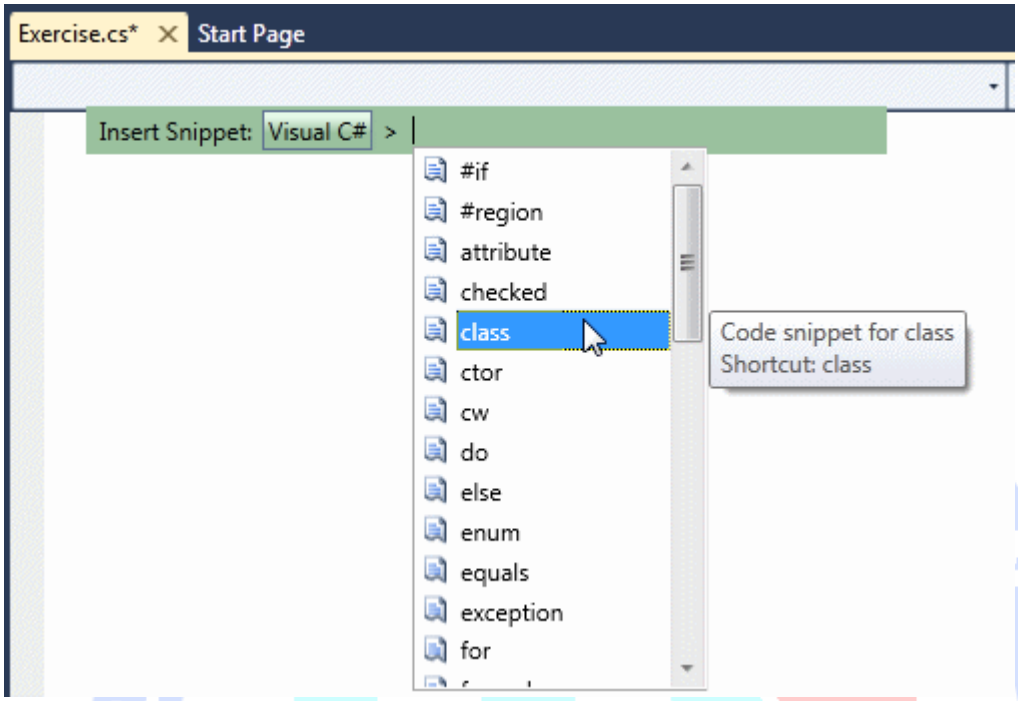
در Solution Explorer، روی اسم پروژه راست کلیک کرده، نشان گر موس را روی Add قرار دهید، سپس class را انتخاب کنید.

در پنجره ی Class View، روی اسم پروژه راست کلیک کنید، نشان گر موس را روی Add قرار داده، و class را انتخاب کنید.

در نتیجه ی تمام عملیات بالا، پنجره ی محاوره ی Add New Item نمایش داده می شود (که در آن گزینه ی class انتخاب شده). در صورتی که Add New Item را از main menu یا Solution Explorer انتخاب کرده اید، کافی است روی خود class کلیک کنید. می توانید اسم پیش فرض را در کادر متن Name عوض کنید (یا در صورت تمایل همان اسم را بپذیرید). اکنون، فایل جدیدی با اسم انتخابی شما و پسوند .cs به پروژه ی شما اضافه می شود.

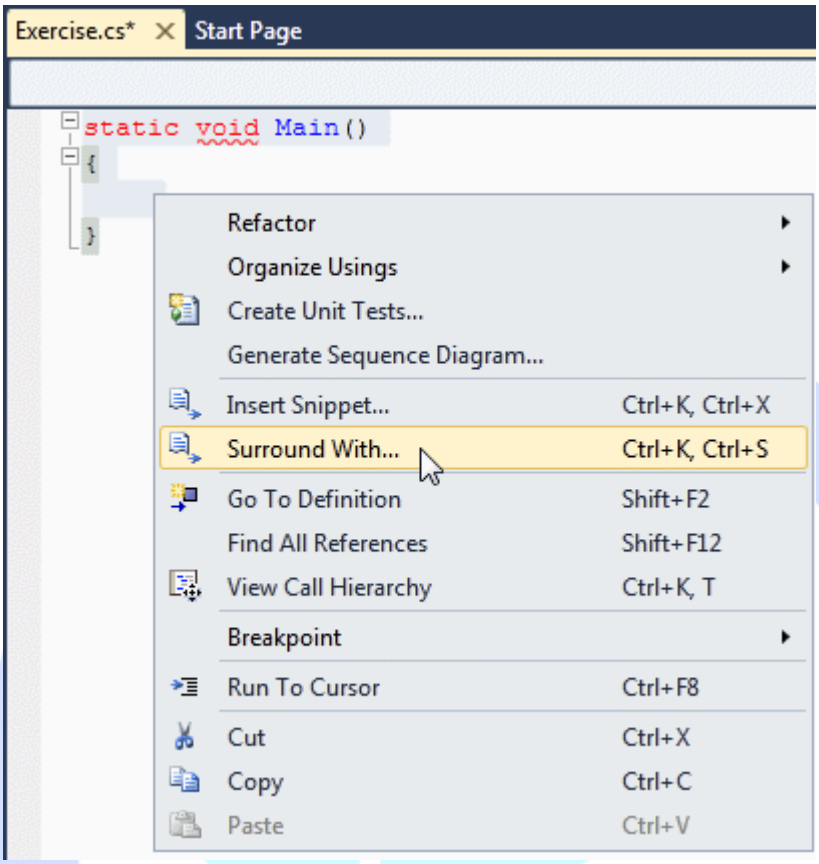
همچنین می توان برای ایجاد کلاس از skeleton code استفاده کرد.

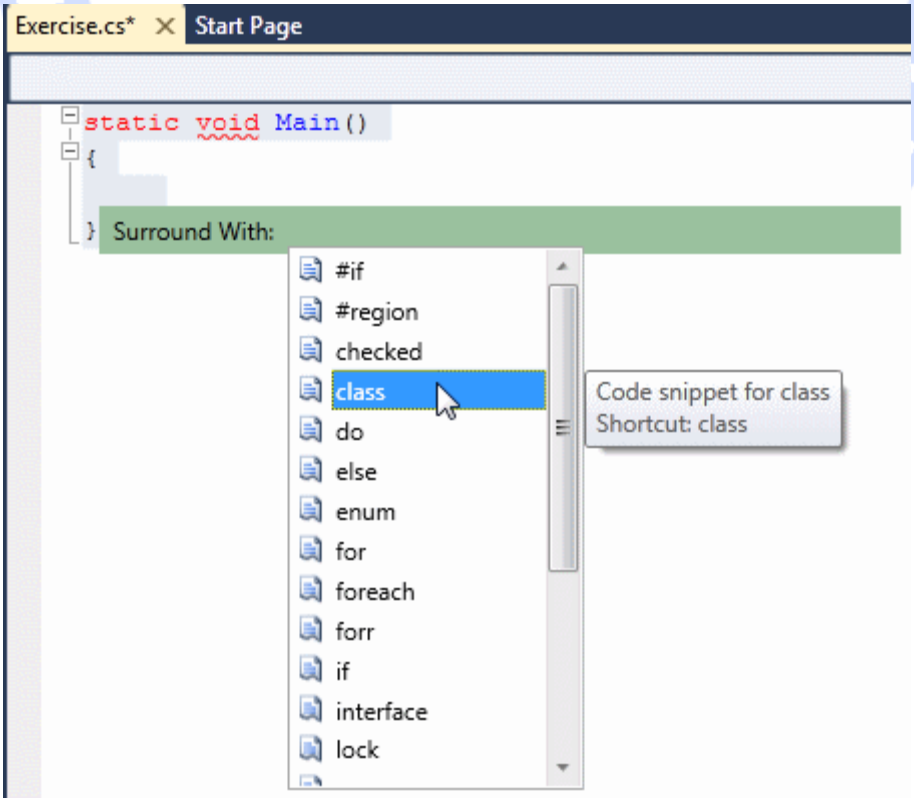
در آن قسمت Code Editor که باید کلاس جدید را اضافه کنید، راست کلیک کرده، سپس روی Insert Snippet کلیک کنید. حال، Visual C# را دو بار کلیک کنید. در لیستی که ظاهر می شود، روی class دو بار کلیک کنید.



چنانچه کد را از قبل نوشته اید، و اکنون می خواهید آن کد را در کلاس خود جای دهید، کافی است آن کد را انتخاب کنید. ابتدا روی آن راست کلیک کرده، سپس گزینه ی Surround With را انتخاب کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330





آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

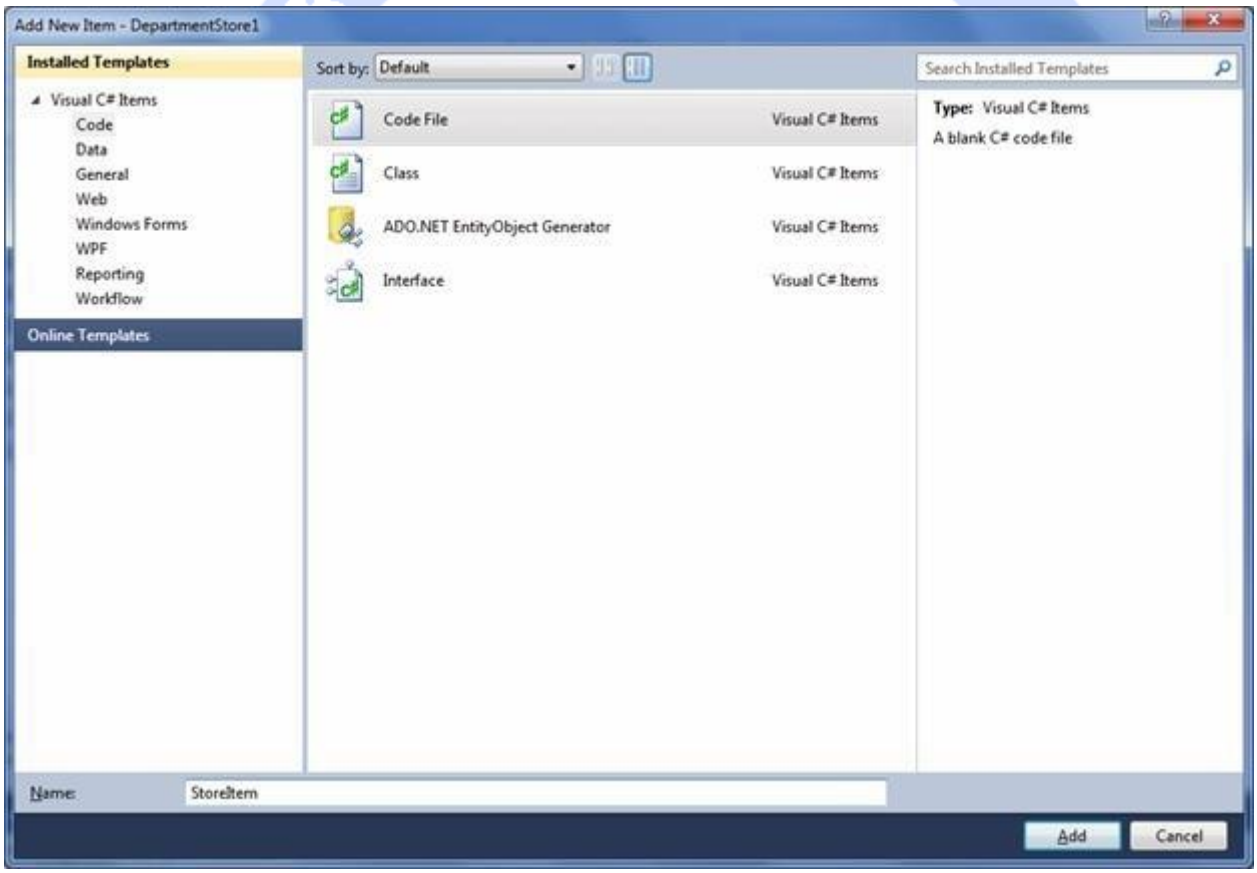
کلاسی با اسم پیش فرض MyClass به وجود می آید.

وارد کردن کلاس

۱. به main menu مراجعه کرده، روی Project -> Add New Item… کلیک کنید.

۲. در لیست میانی Add New Item، روی Code File کلیک کنید.

۳. اسم را به StoreItem تغییر دهید.



۴. روی Add کلیک کنید.

۵. در فایل زیر، این عبارت را تایپ کنید.

```
class StoreItem
{
}
```

۶. در main menu، گزینه های View -> Solution Explorer.

۷. حال، برای به وجود آوردن code file ای دیگر، (در Solution Explorer) روی DepartmentStore1 راست کلیک کرده، سپس نشانگر موس را روی Add قرار داده و گزینه ی click New Item... را انتخاب کنید.

۸. در لیست میانی پنجره ی محاوره Add New Item، روی Code File کلیک کنید.

۹. اسم مورد نظر را به DepartmentStore تغییر دهید.

۱۰. روی Add کلیک کنید.

۱۱. داخل فایل خالی عبارات زیر را تایپ کنید.

```
class DepartmentStore
{
  static void Main()
  {
  }
}
```

نحوه ی دستیابی به کلاس

همان طور که پیشتر ذکر شد، شما می توانید هر تعداد کلاس که مایلید برای پروژه ی خود ایجاد کنید. مطابق میل می توانید هر کلاس را در فایل مختص به خود آن کلاس نگه دارید یا می توانید تمام کلاس ها را در یک فایل واحد ذخیره کنید. اکنون، برای دسترسی به کلاس ها

در قسمتی که به تب های Code Editor اختصاص دارد، روی تَبی که میزبان فایل کلاس دلخواه است کلیک کنید. سپس، به combo box سمت چپ Code Editor مراجعه کرده و کلاس را انتخاب کنید

در Solution Explorer، زیر اسم پروژه، روی اسم فایلی که دربردارنده ی کلاس است دوبار کلیک کنید. چنانچه فایل حاوی چند کلاس است،در combo box سمت چپ Code Editor، کلاس دلخواه را انتخاب کنید.

در پنجره ی Class View، زیر اسم پروژه، اسم کلاس را دوبار کلیک کنید.

در combo box پنجره ی Class View، روی کلاس مورد نظر راست کلیک کرده، سپس گزینه ی Go To Definition را انتخاب کنید.

تغییر اسم کلاس

اگر اسم کلاسی باب میلتان نیست، می توانید آن را عوض کنید. راه اولیه ی آن پیدا کردن اسم کلاس در Code Editor (یا در برنامه ی ویرایش متن مثل Notepad) و ویرایش آن است. در صورت استفاده از این روش، باید تمامی قسمت هایی که اسم در آن به کار رفته نیز پیدا شود. همان طور که انتظار می رود این کار بسیار دشوار بوده و احتمال خطا نیز در آن بالا است. اما برنامه های Microsoft Visual C# 2010 Express و یا Microsoft Visual Studio راه حل بهتری برای انجام این فرایند در اختیار شما قرار می دهند.

برای تغییر دادن اسم در Microsoft Visual Studio یا Microsoft Visual C# دستورات زیر را انجام دهید.

در Code Editor، روی اسم کلاس راست کلیک کرده، حال نشانگر موس را روی Refactor قرار دهید، سپس گزینه ی Rename… را انتخاب کنید.

در پنجره ی Class View، روی اسم کلاس راست کلیک کرده و Rename… را انتخاب کنید.

در Class View، به منظور انتخاب کلاس موردنظر باید روی اسم آن کلیک کنید. اکنون، به main menu مراجعه کرده و گزینه های Refactor -> Rename… را کلیک کنید.

کلیه ی اقدامات بالا پنجره ی محاوره ی Rename را نمایش می دهند. حال، می توانید اسم کلاس را در کادر متن New Name ویرایش کنید. برای تنظیم اینکه آیا مایلید Code Editor، اسم به کار رفته در comment یا string را تغییر دهد یا نه، مجبورید از checkbox ها استفاده کنید. پس از کلیک روی دکمه ی ok، پنجره ی Preview Changes – Rename به نمایش گذاشته می شود. پنجره ی بالا بخش های مختلفی که اسم مورد نظر در آن به کار رفته را برای شما فهرست می کند. اکنون می توانید به منظور اعمال تغییرات به وجود آمده روی Apply کلیک کنید.

ابزار جانبی زبان C#

کد نا امن

یکی از دلایلی که زبان C# طراحی شد، اجتناب از کاستی ها و مشکلات زبان های C/C++ بود. کاربرد اشاره گر (pointer) از جمله ی این کاستی ها بود. C/C++ از اشاره گر برای ارجاع دادن به محل نگه داری مقداری در حافظه استفاده می کنند. بر خلاف این دو زبان، C# تا حد ممکن از به کاربردن اشاره گر اجتناب می کند و مدیریت حافظه را خود به دست می گیرد (به جای سپردن این وظیفه به برنامه نویس). البته در صورت لزوم، برنامه نویس می تواند از اشاره گر (در C#) استفاده کند.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

به این خاطر که compiler زبان C# عهده دار مدیریت حافظه و مقدارهای ذخیره شده در آن است (منظور مقدارهای برنامه ای است که از حافظه استفاده می کنند)، استفاده از اشاره گر توصیه نمی شود. با این وجود، چنانچه مایل هستید از اشاره گر در برنامه ی کاربردی خود استفاده کنید، باید پیش از اسم تمامی متدهایی که از کدناامن استفاده می کنند کلیدواژه ی unsafe را تایپ کنید. به مثال زیر توجه کنید.

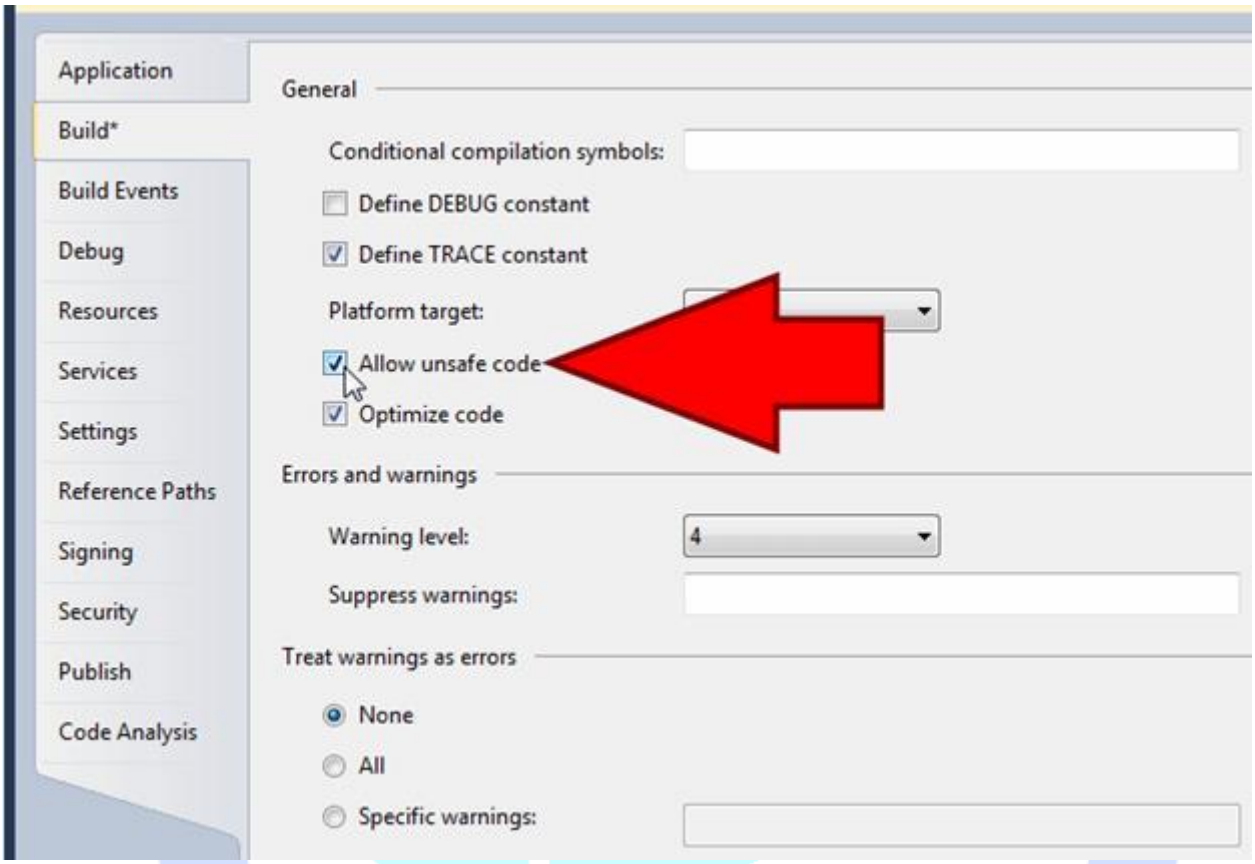
```
class Exercise
{
  unsafe static void Main()
  {
    int Length = 224;
    int *Len = &Length;
    System.Console.Write("Length ");
    System.Console.WriteLine(Length);
    System.Console.Write("Length ");
    System.Console.WriteLine(*Len);
    System.Console.WriteLine();
    Length = 804;
    System.Console.Write("Length ");
    System.Console.WriteLine(Length);
    System.Console.Write("Length ");
    System.Console.WriteLine(*Len);
  }
}
```

پیش از ترجمه (compile) برنامه ی کاربردی مورد نظر، باید حتماً اشاره شود که unsafe code (کد ناامن) در این برنامه به کاربرده شده. برای این منظور، از تعریف کننده (modifier)، unsafe/ استفاده می شود.

نمونه

```
csc /unsafe Exercise.cs
```

برای استفاده از این امکان در Microsoft Visual Studio، main menu را باز کرده، روی ...Project -> [Project] Properties کلیک کنید. حال، در قسمت Build، کادر تیک Allow Unsafe Code را فعال کنید.



Region Delimiters

Microsoft Visual Studio امکانات مختلفی برای نوشتن و مدیریت بهتر کد فراهم می کند. این ویژگی ها و امکانات عبارت اند از : کلمات رنگی، توگذاری خودکار (intuitive indentation)، IntelliSense، تعیین حد و مرز کدها و غیره...

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
Circle
class Circle
{
}

class Rectangle
{
}

class Square
{
}

class Line
{
}
```

همان طور که در تصویر مشاهده می کنید، دکمه هایی در سمت چپ برخی از خط های برنامه قرار دارند. دکمه های نام برده به شما اجازه می دهند آن بخشی از کد را که مورد نیاز نیست جمع کنید (ببندید). برای این منظور، باید روی دکمه ی (ـ) کلیک کنید. علامت دکمه ی مذکور به (+) تغییر می کند.

```
Circle
class Circle
{
}

class Rectangle...

class Square
{
}

class Line
{
}
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

دکمه ی + به شما اجازه می دهد بخش پنهان کد را باز کنید. این ویژگی یکی از امکانات Code Editor برنامه ی Microsoft Visual Studio است (که البته در خیلی از محیط های برنامه نویسی دیگر نیز کاربرد دارد). برای به وجود آوردن این بخش ها، Code Editor از قوانین خاصی پیروی می کند.

لازم به ذکر است که شما می توانید در صورت تمایل، به جای استفاده از بخش های کد بالا، بخش های کد (code section) دلخواه خود را ایجاد کنید. برای این منظور

در صورت کار با برنامه های ویرایش متن (مثل wordpad) یا Code Editor برنامه ی Microsoft Visual C# 2010 Express، فایل موردنظر را با عبارت زیر شروع کرده.

```
#region Whatever
```

و با این عبارت به پایان برسانید.

```
#endregion Whatever
```

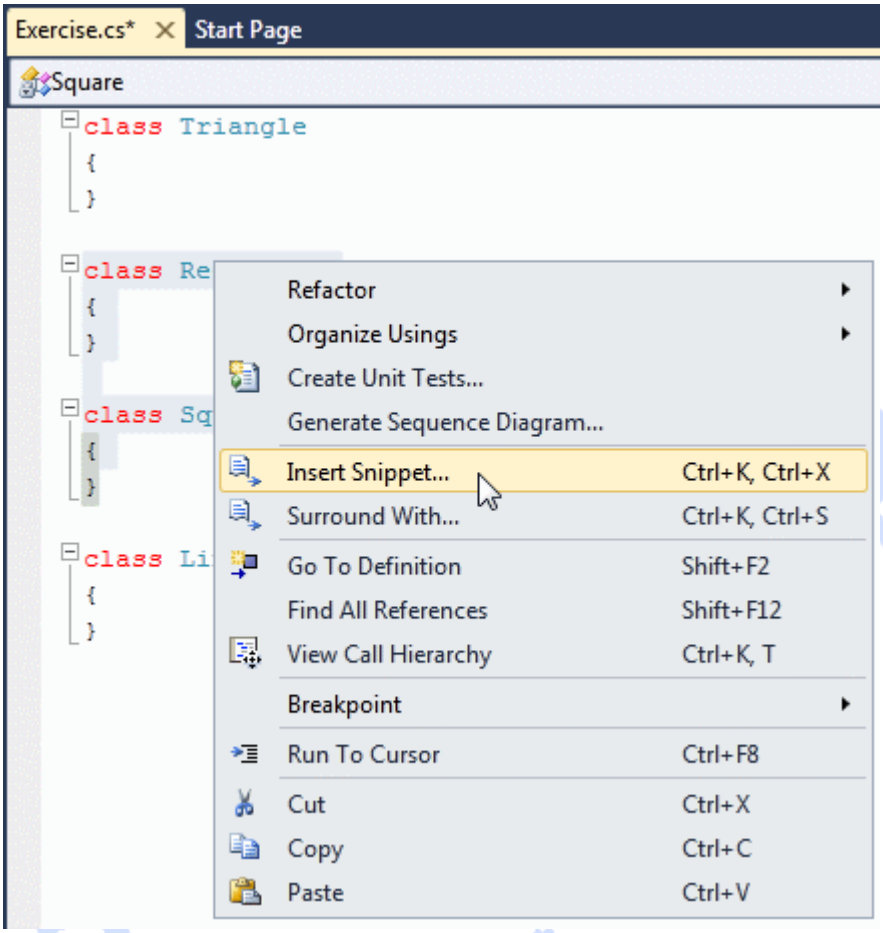
اگر با برنامه های Microsoft Visual C# 2010 Express یا Microsoft Visual Studio کار می کنید.

چنانچه مایلید section ای ایجاد کنید که حاوی بخشی تهی باشد، روی آن خطی که می خواهید section از آن جا شروع شود راست کلیک کرده، Insert Snippet... را انتخاب کنید، سپس

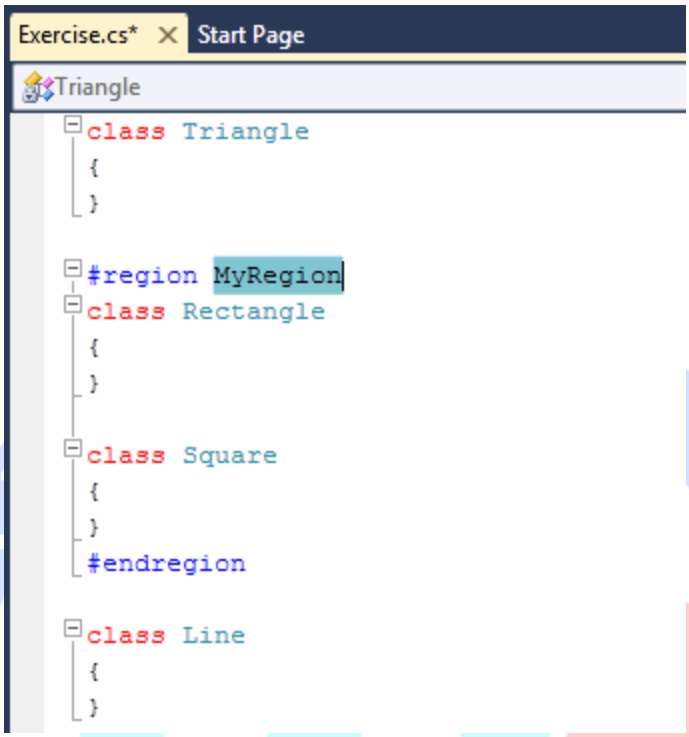
روی Visual C# دوبار کلیک کنید.

چنانچه مایلید بخشی ایجاد کنید که حاوی کد است، کافی است کد موردنظر را در Code Editor انتخاب کنید. حال، باید روی بخش انتخابی راست کلیک کرده و گزینه ی Insert Snippet را انتخاب کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



روی صفحه ی Visual C# دوبار کلیک کنید. در لیستی که ظاهر می‌شود، روی گزینه ی #region کلیک کنید. ناحیه ای با اسم پیش فرض به صورت زیر ایجاد می شود.



هر زمان و جایی که می خواهید ناحیه ی جدیدی ایجاد کنید، به عبارت #region نیاز دارید. می توانید در سمت راست عبارت بالا هر چه مایل هستید تایپ کنید. بخش به وجود آورده را با عبارت #endregion (و هرچیز دیگری که دوست دارید به دنبال آن تایپ کنید) به پایان برسانید. توجه خود را به مثال زیر جلب کنید.

```
class Circle
{
}
#region This section is reserved for quadrilateral shapes
class Rectangle
{
}
class Square
{
}
#endregion This is the end of the quadrilateral section
class Line
{
}
```

نیاز به تایپ کردن هیچ چیز در سمت راست #endregion نیست. پس از ایجاد ناحیه ی مورد نظر، Code Editor دکمه ی ـ را در سمت چپ #region نمایش می دهد و به دنبال آن خطی ظاهر می شود که به سمت چپ عبارت #endregion ختم می شود.

```
Line
class Circle
{
}

#region This section is reserved for quadrilateral shapes
class Rectangle
{
}

class Square
{
}
#endregion This is the end of the quadrilateral section

class Line
{
}
```

حال، این دکمه به شما اجازه می دهد بخش مورد نظر را به دلخواه بازکرده یا ببندید.

```
Line
class Circle
{
}

This section is reserved for quadrilateral shapes

class Line
{
}
```

همان طور که بالا ذکر شد، نیازی نیست در سمت راست #endregion چیزی بنویسید و می توانید آن را خالی بگذارید. در مثال بالا، مستطیلی مشاهده می کنید که با خط های خاکستری رشته را احاطه کرده و به دنبال #region می آید. این مستطیل رشته ای که پس از #endregion می آید را تحت پوشش قرار نمی دهد. به عبارت دیگر، چنانچه هیچ چیز در سمت راست #endregion تایپ نشود، بخش سمت راست خط #region نیز نمایش داده نخواهد شد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

مبانی نحوه ی استفاده از کلاس

ایجاد (یک) شی

برای به کاربردن کلاس در برنامه، ابتدا لازم است برای آن یک متغیر تعریف کنید. از فرایند تعریف متغیر برای کلاس، به عنوان ایجاد شی (creating an object) یا نمونه ای از کلاس (instance of class) نیز یاد می شود.

به منظور تعریف متغیر (یک) کلاس، می توان از کلید واژه ی var استفاده کرد. سرانجام، می توان اسم متغیر را به دنبال اسم کلاس مورد نظر تایپ کرد. برای مثال، اگر بخواهیم متغیری برای کلاس خانه ی اول درس معرفی کنیم، به صورت زیر عمل می کنیم :

```
class House
{
}

class Program
{
  static void Main()
{
  var property . . .
}
}
```

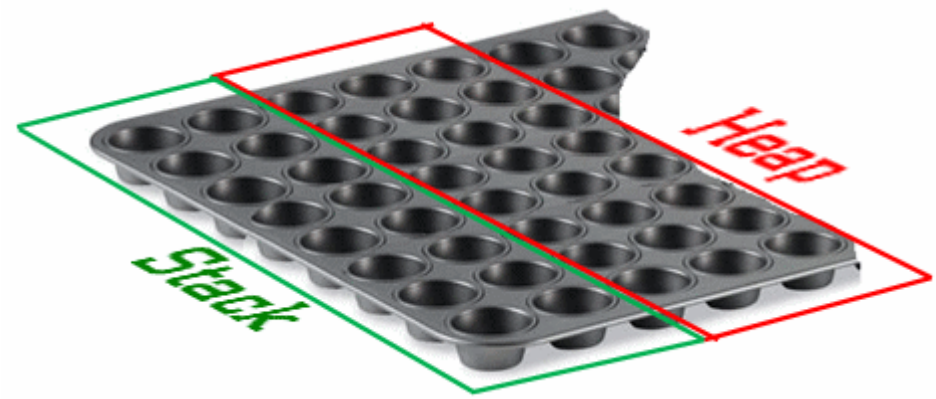
یا

```
class House
{
}
class Program
{
  static void Main()
  {
    House property;
  }
}
```

نوع مقدار (value types) و نوع ارجاع (reference type)

در درس های ۳ و ۴ با نحوه ی تعریف متغیر های نوع عادی آشنا شدیم. همچنین گفتیم که پس از مقداردهی اولیه ی این متغیرها، compiler مقدارهای آن ها را در قسمت هایی از حافظه که از قبل به آن ها اختصاص داده ذخیره می کند. نوع داده (data type) هایی که برای چنین

متغیرهایی به کار می روند، نوع مقدار (value type) خوانده می شوند. برای ذخیره سازی مقادیر این متغیرها، compiler بخشی در حافظه به وجود می آورد به نام پشته (stack). بخش باقی مانده ی حافظه heap نام دارد که به ذخیره سازی متغیر کلاس ها اختصاص می یابد :



پس از این که متغیری را با استفاده از کلاس تعریف کردید، باید از compiler درخواست کنید بخشی از حافظه ی heap را به آن تخصیص دهد. برای این منظور، از عملگری به نام new استفاده می شود.

مثال

```
class House
{
}
class Program
{
  static void Main()
  {
    var property = new House();
  }
}
```

می توان به جای استفاده از کلید واژه ی var، متغیر را با اسم کلاس معرفی کرد. همان طور که برای نوع مقدار (value type) ذکر شد، در صورت استفاده از کلیدواژه ی var، ملزم هستید بلافاصله حافظه به آن اختصاص دهید. اما، در صورت استفاده از اسم کلاس (برای تعریف متغیر) ، می توان متغیر مزبور را در یک خط تعریف کرد، سپس عملگر New را در خط دیگری برای تخصیص حافظه به کاربرد.

مثال

```
class House
{
}
class Program
```

```
{
  static void Main()
  {
    House property;
    // You can do something here
    property = new House();
  }
}
```

در برنامه های Visual Basic و C# چنانچه کلاسی را در هریک از فایل های متعلق به یک پروژه واحد ایجاد کنید، کلاس نام برده در دسترس تمامی فایل های (دیگر) پروژه قرار می گیرد.

تعریف متغیر نوع کلاس

در بدنه ی Main، عبارات زیر را تایپ کنید.

```
class DepartmentStore
{
  static void Main()
  {
    StoreItem si = new StoreItem();
  }
}
```

ایجاد شی تهی (Null Object)

برای استفاده از امکانات (behavior) و مقدارهایی که کلاس فراهم می کند، ملزم به تعریف متغیر هستیم. همچنین دیدیم که پیش از به کاربری کلاس، باید مقدارهای لازم برای فیلد های کلاس را فراهم کنید. این کار به compiler اجازه می دهد تا حافظه ی تخصیص یافته به متغیر را با مقادیر دلخواه پر کند. ممکن است شما کاملاً آماده نباشید مقادیر لازم را هنگام تعریف متغیر فراهم کنید.

متغیری که برای کلاسی تعریف شده ولی هنوز مقدار لازم برای فیلد را دریافت نکرده را، شی تهی (null object) می خوانند. برای این منظور (ایجاد شی تهی)، ابتدا اسم کلاس و به دنبال آن اسم متغیر را تایپ کرده، سپس کلید واژه ی null را به آن اختصاص دهید.

پیش تر بحث شد که می توان برای تعریف متغیری با مقدار تهی (null) علامت سوال (؟) به آن اضافه کرد. در صورت استفاده از کلاس، از اضافه کردن ؟ به اسم کلاس خودداری کنید.

مثال

```
class House
{
```

```
}
class Program
{
  static void Main()
  {
    House property = null;
  }
}
```

چنانچه متغیر کلاسی از پیش تعریف و مقدار دهی اولیه شده بود، ولی بنا به هر دلیلی مقدارهای آن پاک شد، چنین متغیری نیز شی تهی (null object) خوانده می شود.

به اشتراک گذاری کلاس

برخلاف زبان های C و C++ که پیشرو و والدین برنامه ی C# حساب می شود، زبان C# طوری تعبیه شده که با دیگر زبان های برنامه نویسی از جمله C++/CLI، Visual Basic و F# به صورت مکمل کار کند. به عبارت دیگر، می توان با کد هریک از زبان های بالا، کدهای یک برنامه کاربردی که توسط C# نوشته شده باشد را خواند. برای تحقق بخشیدن به این امر، کلاس C# باید به عنوان شی عمومی (public object) تعریف شود.

اگر می خواهید کدهای نوشته شده توسط زبان های بالا به کلاس شما دسترسی پیدا کنند، باید هنگام ایجاد کلاس، پیش از کلید واژه ی class، کلیدواژه ی public را تایپ کنید.

مثال

```
public class Exercise
{
  static void Main()
  {
    var number = 244;
    var thing = "Vehicle";
    System.Console.WriteLine(number);
    System.Console.WriteLine(thing);
  }
}
```

کاربرد : به اشتراک گذاری کلاس

۱. main menu را باز کرده، روی گزینه های View -> Class View کلیک کنید.

۲. در پنجره ی Class View، روی DepartmentStore1 دوبار کلیک کنید تا بزرگ شود.

۳. در قسمت بالایی بدنه ی اصلی پنجره ی Class View، روی StoreItem راست کلیک کرده، سپس گزینه ی Go To Definition را انتخاب کنید.

۴. کلید Home را زده، public را تایپ کنید و پس از آن دکمه ی space را فشار دهید. فایل پیش رو به صورت زیر تغییر داده می شود.

```
public class StoreItem
{
}
```

زباله روبی (garbage collection)

زمانی که متغیری را به کمک کلیدواژه ی new تعریف می کنید، در واقع دارید بخشی از حافظه ی heap را به آن اختصاص می دهید (برای آن رزرو می کنید). زمانی که به آن متغیر دیگر نیازی نیست، مثل زمانی که برنامه بسته می شود، متغیر گفته شده پاک شده و حافظه ی اختصاص یافته به آن باید در اختیار متغیرها یا برنامه های دیگر قرار گیرد. به این فرایند زباله روبی یا garbage collection می گویند. در زبان های برنامه نویسی C/C++، این مسئله برای برنامه نویس یک معضل بزرگ محسوب می شد، زیرا که برنامه نویس باید خود به صورت دستی چنین متغیرهایی (اشاره گر/pointer) را پاک می کرد و حافظه را آزاد می ساخت.

چهارچوب کاری .NET Framework مسئله ی زوباله روبی را حل کرده و حافظه را خودکار از متغیرهای بلااستفاده رها می سازد.

فیلدهای کلاس

کلاسی با اسم House را در نظر بگیرید.

```
public class House
{
}
```

به بخشی که محصور در علامت {} است، بدنه ی یک کلاس گفته می شود. در بدنه ی کلاس اجزای تشکیل دهنده ی کلاس قرار می گیرند. این اجزا در واقع تحت عنوان متغیرهای عضو یا member variable های یک کلاس تعریف می شوند. در زبان C# متغیری که در بدنه ی کلاس ایجاد می شود، فیلد اطلاق می گردد. هر فیلد مثل متغیرهای معمول دیگر، با یک اسم و نوع داده تعریف می شود.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

برای مثال، زیر اجزای تشکیل دهنده ی یک خانه به عنوان اعضای تشکیل دهنده یک کلاس (که هریک خود به مثابه ی یک متغیر تعریف شده) تعریف شده اند.

```
public class House
{
  string propertyNumber;
  char propertyType;
  byte stories;
  uint bedrooms;
  decimal value;
}
```

برای این که نشان دهید متغیری (فیلدی از نوع اولیه) قادر است مقداری تهی (null value) داشته باشد، کافی است علامت سوال به آن اضافه کنید.

```
public class House
{
  string propertyNumber;
  char? propertyType;
  byte? stories;
  uint? bedrooms;
  decimal? value;
}
```

پس از افزودن ؟، ملزم به رعایت قوانین خاصی می شوید.

متغیرهای تعریف شده در بدنه ی یک کلاس متغیرهای عضو (member variable) خوانده می شوند و هر متغیر عضو سرانجام یک فلید تلقی می گردد. فیلدها ی یک کلاس ممکن از هر نوع داده که تا کنون معرفی شده تشکیل شده باشد.

توجه

از این بخش به بعد، برای اشاره به فیلد متعلق به یک کلاس می نویسیم.

ClassName.FieldName

به عبارتی ساده تر، FieldName فیلدی است که عضو کلاس ClassName می باشد.

افزودن فیلد به کلاس

کلاس StoreItem را به این ترتیب اصلاح کنید.

```
public class StoreItem
{
  long itemNumber;
  string itemName;
  string size;
  decimal unitPrice;
}
```

سطح دسترسی اعضای کلاس (access modifiers of a class)

اعضای خصوصی یک کلاس

بخش های یک شئ به دو دسته ی اصلی تقسیم می شوند : آن دسته ای که به آن دسترسی دارید و آن دسته ای که دسترسی به آن ها برای شما مقدورنمی باشد. برای مثال، ماشینی که در خیابان پارک شده را می توان دید، آن را لمس کرد ولی موتور آن برای شما قابل رویت و دسترسی نیست. به اجزایی از کلاس که در دسترس عموم قرار دارند public گفته می شود.

اعضای عمومی یک کلاس

بخش هایی از کلاس که از دید و دسترسی عموم پنهان است private تلقی می گردد. در زبان C# برخی اعضای کلاس در اختیار دیگر کلاس ها قرار می گیرند و در عین حال برخی دیگر اعضای کلاس از دید و دسترس کلاس های دیگر پنهان می شوند. برای محدود کردن سطح دسترسی به بخش یا عضوی از کلاس که باید مخفی یا خصوصی باقی بماند از کلید واژه ی private استفاده می شود. حال، برای در دسترس قرار دادن آن بخش از کلاس که خصوصی محسوب نمی شود، باید متغیر را با کلید واژه ی public تعریف کرد. به کلید واژه های public و private سطح دسترسی (access level) نیز گفته می شود.

به صورت پیش فرض، تمامی متغیر های عضوی که تعریف می کنید ولی سطح دسترسی آن را مشخص نمی کنید، private محسوب می شوند و بدین ترتیب برای غیراعضای یک کلاس قابل دسترسی نمی باشد. به این خاطر چنانچه می خواهید عضوی را برای غیراعضای یک کلاس قابل دسترس کنید، باید آن را با کلید واژه ی puplic تعریف کنید. می توان ترکیبی از اعضای خصوصی و عمومی در کلاس خود ایجاد کرد و هیچ قانون خاصی مبنی بر این که کدام اول باید فهرست شود و کدام آخر وجود ندارد.

مثال

```
public class House
{
  string propertyNumber;
  public char propertyType;
```

```
    byte stories;
    public uint bedrooms;
    private decimal value;
}
```

فقط به خاطر داشته باشید که در صورت تعیین نکردن سطح دسترسی، عضو کلاس تعریف شده خودکار خصوصی یا private می شود. برای جلوگیری از بروز هر نوع خطا، حتماً توصیه می شود سطح دسترسی عضو کلاس را مشخص کنید.

```
public class House
{
    public string propertyNumber;
    public char propertyType;
    public byte stories;
    public uint bedrooms;
    public decimal value;
}
```

تعیین سطح دسترسی به اعضای کلاس

کلاس storeitems را به صورت زیر اصلاح کنید.

```
public class StoreItem
{
    public long itemNumber;
    public string itemName;
    public string size;
    public decimal unitPrice;
}
```

اعضای داخلی کلاس

دیدیم که کلیدواژه ی public به اشیاِی خود برنامه و برنامه های دیگر اجازه می دهد که به عضو عمومی یک کلاس دسترسی پیدا کنند. در حالی که با استفاده از کلید واژه private تنها به اعضای خود آن کلاس اجازه ی دسترسی به اعضای خصوصی را فراهم می کردیم. چنانچه مایل هستید عضوی ایجاد کنید که تنها برای اعضا و اشیاِ خود آن کلاس قابل دسترسی باشد، لازم است آن را با کلید واژه ی internal علامت گذاری کنید. تفاوت بین این کلید واژه ها در زیر شرح داده شده.

چنانچه کلاسی با کلیدواژه های زیر علامت گذاری یا تعریف شده باشد :	

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

private	internal	public	
بله	بله	بله	فقط دیگر اعضای همین کلاس می توانند به این عضو دسترسی پیدا کنند
خیر	بله	بله	تمام اعضای این برنامه، حتی اعضای خارج از این کلاس، به این عضو دسترسی دارند
خیر	خیر	بله	اشیا خارج از این برنامه می توانند به این عضو دسترسی پیدا کنند

مقدار دهی اولیه ی شی

عملگر نقطه (.)

به منظور دستیابی به عضوی از کلاس از طریق کلاسی دیگر، ابتدا باید متغیری از آن کلاس را تعریف کنید. اکنون، برای دسترسی به عضو موردنظر از عملگر "." استفاده کنید.

پس از تعریف نمونه ای از کلاس، می توان به اعضای کلاس نام برده دسترسی داشت و به آن ها مقادیر دلخواه را تخصیص داد. به مثال زیر توجه کنید.

```
public class House
{
  internal long propertyNumber;
  internal string propertyType;
  internal uint bedrooms;
  internal double value;
}

public class Exercise
{
  static void Main()
  {
    var property = new House();
    property.propertyNumber = 283795;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    property.propertyType = "Single Family";
    property.bedrooms = 4;
    property.value = 652880;
  }
}
```

به خاطر دارید که داده های نوع اولیه (primitive types) می توانند علامت سوال داشته باشند.

```
public class House
{
  internal long? propertyNumber;
  internal string propertyType;
  internal uint? bedrooms;
  internal double? value;
}
public class Exercise
{
  static void Main()
  {
    var property = new House();
    property.propertyNumber = 283795;
    property.propertyType = "Single Family";
    property.bedrooms = 4;
    property.value = 652880;
  }
}
```

پس از مقدار دهی اولیه ی فیلد یک کلاس، می توان با به کار بردن عملگر "." به آن دسترسی پیدا کرده، سپس مقدار آن را بازیابی کرد.

```
public class House
{
  internal long? propertyNumber;
  internal string propertyType;
  internal uint? bedrooms;
  internal double? value;
}
public class Exercise
{
  static void Main()
  {
    var property = new House();
    property.propertyNumber = 283795;
    property.propertyType = "Single Family";
    property.bedrooms = 4;
    property.value = 652880;
    System.Console.WriteLine("=//= Altair Realtors =//=");
    System.Console.WriteLine("Properties Inventory");;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    System.Console.Write("Property #: ");
    System.Console.WriteLine(property.propertyNumber);
    System.Console.Write("Property Type: ");
    System.Console.WriteLine(property.propertyType);
    System.Console.Write("Bedrooms:    ");
    System.Console.WriteLine(property.bedrooms);
    System.Console.Write("Market Value: ");
    System.Console.WriteLine(property.value);
    System.Console.ReadKey();
  }
}
```

نتیجه ی زیر را به دست می دهد.

```
=//= Altair Realtors =//=
Properties Inventory
Property #: 283795
Property Type: Single Family
Bedrooms:    4
Market Value: 652880
Press any key to continue...
```

به کاربردن فیلدهای یک کلاس

۱. در بالاترین قسمت Code Editor، روی DepartmentStore.cs کلیک کرده تا به فایل دسترسی پیدا کنید.

۲. فایل بالا را بدین ترتیب اصلاح کنید.

```
class DepartmentStore
{
  static void Main()
  {
    StoreItem si = new StoreItem();
    si.itemNumber = 720823;
    si.itemName = "Cotton Seam Sheath Dress";
    si.size = "6";
    si.unitPrice = 158M;
    System.Console.WriteLine("Department Store");
    System.Console.Write("Item #:   ");
    System.Console.WriteLine(si.itemNumber);
    System.Console.Write("Item Name: ");
    System.Console.WriteLine(si.itemName);
    System.Console.Write("Item Size: ");
    System.Console.WriteLine(si.size);
```

```
    System.Console.Write("Unit Price: ");
    System.Console.WriteLine(si.unitPrice);
    System.Console.ReadKey();
  }
}
```

۳. برنامه را اجرا کنید. نتیجه ی زیر حاصل می گردد.

```
Department Store
Item #:  720823
Item Name: Cotton Seam Sheath Dress
Item Size: 6
Unit Price: 158
```

٤. حال، کادر پیغام (message box) را ببندید.

به کاربردن نوع ناشناس

همان طور که پیش تر ذکر شد، برای به کاربردن کلاس ابتدا باید متغیری برای آن تعریف کرده، سپس آن را مقدار دهی اولیه کنید. خوشبختانه به منظور مقدار دهی اولیه یک شی، نیازی به استفاده از کلاس نیست. می توان متغیری مشابه نمونه ی کلاس تعریف کرده، بعد آن را به صورت دلخواه مقدار دهی اولیه کرد. از این نوع با نام anonymous type (یا نوع ناشناس) یاد می شود. به منظور استفاده از آن کافی است متغیر مورد نظر را با کلید واژه ی var تعریف کرده و به منظور تخصیص حافظه به آن از عملگر new استفاده کنید. به جای استفاده از اسم کلاس، ابتدا علامت {} را درج کرده، سپس عملگر new را تایپ کنید. حال در {} اسمی برای تک تک اعضا انتخاب کرده، گویی دارید اسم عضو های کلاس را تعریف می کنید، سپس هر متغیر عضو را با value های دلخواه مقدار دهی اولیه کنید. پس از علامت {}، تعریف را با (;) به پایان برسانید.

مثال

```
public class Exercise
{
  static void Main()
  {
    var BookInformation = new
    {
      Title = "Calculus 6e Edition",
      Pages = 1074,
      Cover = "Hard Back"
    };
  }
```

```
}
```

پس از مقدار دهی اولیه ی نوع ناشناس، می توان با تایپ کردن به ترتیب اسم متغیر، عملگر نقطه، متغیر عضو، به تک تک اعضای آن دسترسی پیدا کرد.

مثال

```
public class Exercise
{
  static void Main()
  {
    var BookInformation = new
    {
      Title = "Calculus 6e Edition",
      Pages = 1074,
      Cover = "Hard Back"
    };
    System.Console.WriteLine("=//= BookInformation =//=");
    System.Console.Write("Title:       ");
    System.Console.WriteLine(BookInformation.Title);
    System.Console.Write("Nbr of Pages: ");
    System.Console.WriteLine(BookInformation.Pages);
    System.Console.Write("Type of Cover: ");
    System.Console.WriteLine(BookInformation.Cover);
  }
}
```

نتیجه ی زیر حاصل می گردد.

```
=//= BookInformation =//=
Title:       Calculus 6e Edition
Nbr of Pages: 1074
Type of Cover: Hard Back
Press any key to continue...
```

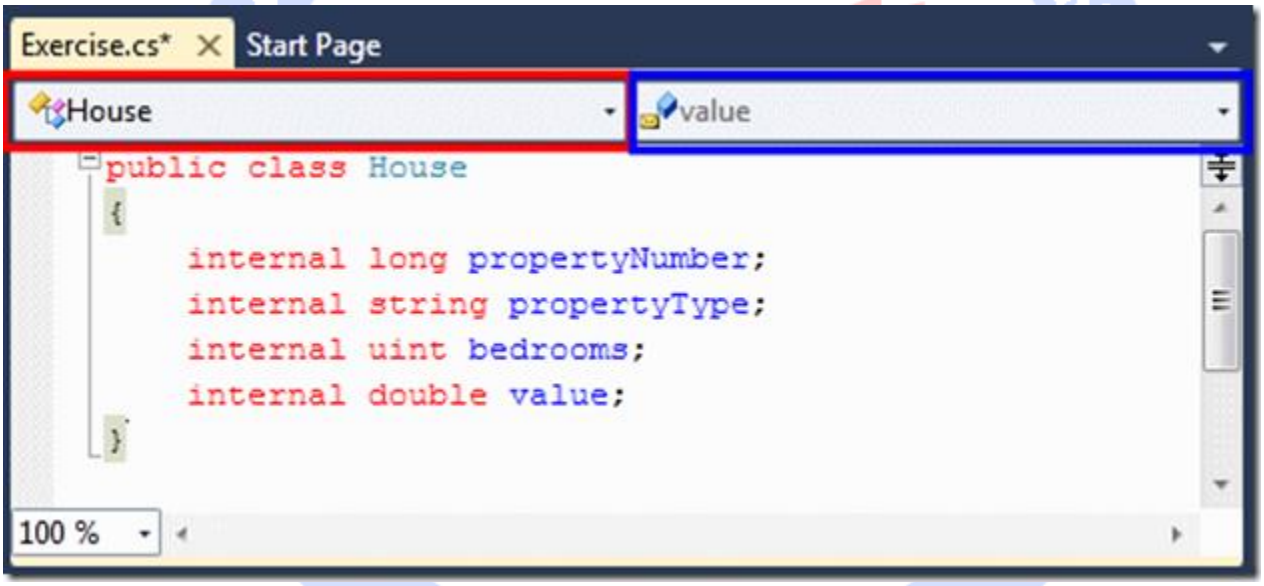
به خاطر داشته باشید که قرار دادن تعریف در خط های متعدد فقط خواندن آن را سهل می کند. می توان تمامی جزئیات را در خطی یکسان قرار داد.

```
public class Exercise
{
  static void Main()
  {
    var book = new { Title = "Calculus 6e Edition", Pages = 1074 };
    System.Console.WriteLine("=//= BookInformation =//=");
```

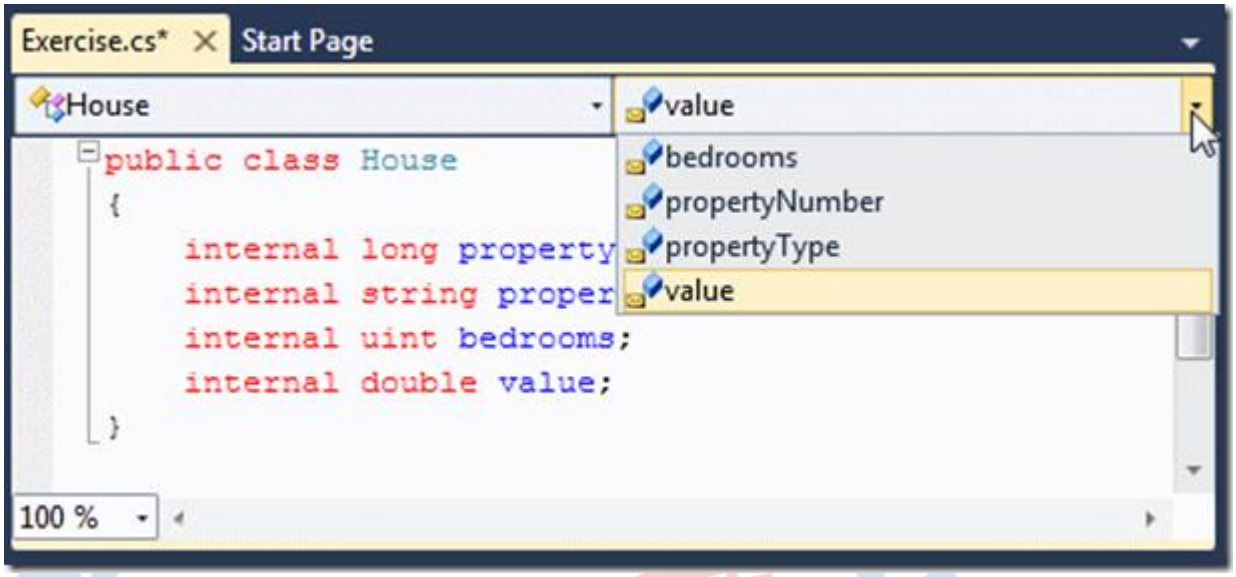
```
    System.Console.Write("Title:      ");
    System.Console.WriteLine(book.Title);
    System.Console.Write("Nbr of Pages: ");
    System.Console.WriteLine(book.Pages);
    System.Console.ReadKey();
  }
}
```

مدیریت فیلدهای کلاس

برای سهولت و مدیریت بهتر فیلدهای یک کلاس، Code Editor از دو combo box بهره می گیرد.



Combo box سمت چپ، کلاس های موجود در فایل جاری را نمایش می دهند. Combo box سمت راست، اعضای انتخابی کلاس از Combo box سمت چپ را به نمایش می گذارد.



نحوه ی دسترسی به فیلد

از راه های مختلفی می توان به فیلد دسترسی پیدا کرد از جمله

فایل دربردارنده ی کلاس را باز کنید. همان طور که مشاهده می کنید، فیلد در فایل بیان شده قابل رویت می باشد

فایل حاوی کلاس را باز کنید. به main menu مراجعه کرده، گزینه های Edit -> Quick Find را انتخاب کنید. اسم فیلد مورد نظر را تایپ کرده، روی Find Next کلیک کنید. لازم به ذکر است که در این روش، باید اسم دقیق فیلد مورد نظر را از قبل داشته باشید.

در قسمت بالایی بدنه ی پنجره ی Class View، روی اسم کلاس حاوی فیلد کلیک کنید. لیست اعضای آن در قسمت پایینی بدنه ی پنجره نمایان می شود. اکنون می توانید روی آن دوبار کلیک کنید.

در بخش فوقانی بدنه ی Class View، روی اسم کلاس حامل فیلد کلیک کنید. در قسمت پایینی Class View، روی اسم فیلد راست کلیک کرده، سپس گزینه ی Go To Definition را کلیک کنید.

تغییر اسم فیلد

می توان اسم فیلد را باب میل تغییر داد. کافی است فیلد را در فایل مورد نظر پیدا کرده و با دانش خود از ویرایش متن (text editing) نام فیلد را اصلاح کرد. سرانجام، می توانید پس از باز کردن فایل حاوی کلاس، در main menu روی Edit -> Quick Replace کلیک کنید، اسم فیلد را تایپ کرده، سپس Find Next یا Replace را انتخاب کنید. این روش برای بازدید کردن از تمام بخش هایی که فیلد در آن به کار رفته بسیار کارامد است.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

محیط برنامه نویسی visual studio برای این امر بسیار مناسب است.

به منظور تغییر اسم در محیط visual studio

فایل حاوی کلاس را باز کنید. یا کلیدهای Ctrl + H را با هم بزنید، و یا در main menu روی گزینه ی Quick Replace کلیک کنید. حال، در قسمت Find What، اسم فیلد را وارد کرده و Replace All را انتخاب کنید.

فایل حامل کلاس را باز کنید. روی فیلد مورد نظر کلیک کنید تا علامت ^ (caret) در آن قرار داده شود. کلیدهای Ctrl + H را بزنید و یا در main menu، Quick Replace را کلیک کنید. combo box ای به نام Find What ظاهر می شود که در آن فیلد انتخابی را وارد کرده، سپس روی دکمه ی Replace All کلیک کنید.

در سمت بالایی پنجره ی Class View، باید روی اسم کلاس کلیک کنید. در قسمت پایینی پنجره، روی فیلد راست کلیک کرده سپس Rename را انتخاب کنید... حال، در کادر متن New Name اسم دلخواه فیلد را درج کنید. برای تنظیم و تغییر اسم در بخش comment ها و string ها، باید به checkbox ها مراجعه کرد. پس از انجام دستورات بالا، روی ok کلیک کنید. در پنجره ی محاوره Preview Changes – Rename، فهرستی از تمامی بخش هایی که فیلد در آن به کار رفته نمایان می باشد. اکنون گزینه Apply را انتخاب کنید.

مبانی متدها

فیلدها در واقع وظیفه ی توصیف کلاس را بر عهده دارند. برای مثال، کلاسی به نام House توسط جنبه هایی همچون تعداد اتاق های خواب (bedrooms)، نوع ملک (property type) و ارزش (value) آن توصیف می شود.

```
public class House
{
  public char propertyType;
  public uint bedrooms;
}
```

علاوه بر این، شی می تواند عملیاتی و یا انتسابی معین انجام دهد. عملیاتی که یک کلاس برای شما انجام می دهد، function یا method خوانده می شود. متد اساساً بخشی از کد است، که در راستای قابلیت ها و کاربرد کلاس، یکی از جزئیات عملیات را ایفا می کند.

معرفی متدها

۱. Microsoft Visual Studio را راه اندازی کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

۲. برای ایجاد برنامه ی کاربردی جدید، به main menu مراجعه کرده و گزینه های Project File -> New انتخاب کنید.

۳. گزینه ی Empty Project را از لیست میانی انتخاب کنید.

۴. اسم را به DepartmentStore2 تغییر داده، سپس ok را کلیک کنید.

۵. در main menu گزینه های Project -> Add New Item... را انتخاب کنید.

۶. در فهرست میانی پنجره ی محاوره ی Add New Item، گزینه ی Code File را انتخاب کنید.

۷. اسم مورد نظر را به StoreItem تغییر داده، بعد Add را کلیک کنید.

۸. در فایل پیش رو دستورات زیر را تایپ کنید.

```
public class StoreItem
{
  public long itemNumber;
  public string itemName;
  public string size;
  public decimal unitPrice;
}
```

۹. به منظور ایجاد code file، در Solution Explorer، روی DepartmentStore2 -> Add -> New Item... راست کلیک کنید.

۱۰. در لیست میانی پنجره ی محاوره Add New Item، Code File را کلیک کنید.

۱۱. اسم را به DepartmentStore تغییر داده و روی Add کلیک کنید.

۱۲. در فایل خالی که در اختیارتان قرار می گیرد، عبارات زیر را تایپ کنید.

```
public class DepartmentStore
{
  static void Main()
  {
    StoreItem si = new StoreItem();
    si.itemNumber = 720823;
    si.itemName = "Cotton Seam Sheath Dress";
    si.size = "6";
    si.unitPrice = 158M;
    System.Console.WriteLine("Department Store");
    System.Console.Write("Item #:  ");
    System.Console.WriteLine(si.itemNumber);
```

```
    System.Console.Write("Item Name: ");
    System.Console.WriteLine(si.itemName);
    System.Console.Write("Item Size: ");
    System.Console.WriteLine(si.size);
    System.Console.Write("Unit Price: ");
    System.Console.WriteLine(si.unitPrice);
    System.Console.ReadKey();
    System.Console.ReadKey();
  }
}
```

۱۳. برای مشاهده ی نتیجه برنامه را اجرا کنید.

۱۴. پنجره ی DOS را بسته و به محیط برنامه نویسی بازگردید.

ساختن متد

برای ایجاد یک متد باید ابتدا اسم آن را مشخص کنید. فرایند تعریف اسم متد نیز از همان قوانینی که برای انتخاب اسم متغیرها تشریح کردیم پیروی می کند. در این مبحث، همان پیشنهاداتی را که برای گزینش اسم کلاس ارائه کردیم، برای انتخاب اسم متد نیز به کار می بریم.

چنانچه اسم متد در یک کلمه خلاصه شده باشد، فقط حرف اول را با حروف بزرگ می نویسیم.

در صورتی که اسم متد متشکل از چند کلمه باشد، حروف اول تمام کلمات با حرف بزرگ نمایش داده می شود.

به این خاطر که متد یک عملیات محسوب می شود، سرانجام باید آن را یک فعل نام گذاشت به دنبال اسم متد پرانتز می آید.

یکی از کارهای متد تخصیص دادن یک مقدار معین در برنامه است. بدین ترتیب پس از این کار، متد می تواند مقداری را فراهم کند. در نتیجه، متد نتیجه را بر می گرداند. چنانچه نتیجه ای از متد نام برده حاصل نشد، به آن void می گویند. نوعی (type) که متد ارائه می دهد (یا باز می گرداند)، در سمت چپ اسم متد درج می شود. چنانچه متد نتیجه ای دربر نداشت، در سمت چپ آن void تایپ کنید. عملیات تخصیص که متد نقل می کند، داخل {} محصور می شود.

نمونه

```
public class House
{
  public char propertyType;
  public uint bedrooms;
  void Display()
  {
  }
}
```

پرکاربرد ترین متد زبان برنامه نویسی C#، متد Main نامیده می شود.

توجه : از این بخش به بعد، برای اشاره به متدی که متعلق به یک کلاس است از *ClassName.MethodName* استفاده می کنیم.

سطح دسترسی متد

درست مانند متغیر عضو اگر قرار است متد تنها برای اعضای خود آن کلاس قابل دسترسی باشد، متد گفته شده را با کلید واژه ی private علامت گذاری کنید (یا می توان هیچ گونه تنظیم کننده ی دسترسی / acces modifier خاصی لحاظ نکرد).

به منظور این که متد برای تمامی اعضای کلاس و دیگر بخش های همین برنامه (نه خارج از این برنامه) قابل دسترسی باشد، آن را با کلید واژه ی internal تعریف کنید.

چنانچه مایل هستید متدی برای تمامی اعضای این کلاس، دیگر بخش های همین برنامه و اجزای برنامه های دیگر قابل دستیابی باشد، کافی است متد بیان شده را با کلید واژه ی public تعریف کنید.

پس از ایجاد متد، می توان به تعریف کارکرد آن (در بدنه ی متد، داخل علامت {}) پرداخت. برای مثال، می توان متغیر های عضو را داخل پرانتزهای System.Console.Write() یا System.Console.WriteLine() نوشت. به مثال های زیر توجه کنید.

```
public class House
{
  public char propertyType;
  public uint bedrooms;
  internal void Display()
  {
    System.Console.WriteLine("=//= Altair Realtors =//=");
    System.Console.WriteLine("Properties Inventory");;
    System.Console.Write("Property Type: ");
    System.Console.WriteLine(PropertyType);
    System.Console.Write("Bedrooms:    ");
    System.Console.WriteLine(Bedrooms);
  }
}
```

می توانید هر تعداد متد که لازم است، به همین ترتیب ایجاد کنید.

فراخوانی متد

پس از ساختن متد، می توان از داخل یا خارج کلاس به آن دسترسی پیدا کرد. فراخوانی یک متد، در واقع همان دسترسی پیدا کردن به متد است. برای این منظور، از عملگر نقطه کمک گرفته می شود. بر خلاف فیلد، لازم است به دنبال اسم کلاس پرانتز گذاشته شود. نمونه ی آن را در مثال زیر مشاهده می کنید.

```
public class House
{
  public char propertyType;
  public uint bedrooms;
  internal void Display()
  {
    System.Console.WriteLine("=//= Altair Realtors =//=");
    System.Console.WriteLine("Properties Inventory");;
    System.Console.Write("Property Type: ");
    System.Console.WriteLine(propertyType);
    System.Console.Write("Bedrooms:    ");
    System.Console.WriteLine(bedrooms);
  }
}
public class Program
{
  static void Main()
  {
    var property = new House();
    property.propertyType = 'S';
    property.bedrooms = 4;
    property.Display();
  }
}
```

نتیجه ی زیر به دست می آید.

```
=//= Altair Realtors =//=
Properties Inventory
Property Type: S
Bedrooms:    4
Press any key to continue...
```

نحوه ی ایجاد متدهای یک کلاس

۱. در main menu، گزینه های Window -> StoreItem.cs را انتخاب کنید. فایل مورد نظر باز می شود.

۲. برای افزودن متد به کلاس، فایل را به صورت زیر اصلاح کنید.

```
public class StoreItem
{
  public long itemNumber;
  public string itemName;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
  public string size;
  public decimal unitPrice;
  public void CreateItem()
  {
    itemNumber = 792475;
    itemName = "Aramis Gentlemen Collection 3.4oz JHL Custom Blended Cologne Spray";
    size = "3.4oz";
    unitPrice = 48.00m;
  }
  public void Describe()
  {
    System.Console.WriteLine("Department Store");
    System.Console.Write("Item #:  ");
    System.Console.WriteLine(itemNumber);
    System.Console.Write("Item Name: ");
    System.Console.WriteLine(itemName);
    System.Console.Write("Item Size: ");
    System.Console.WriteLine(size);
    System.Console.Write("Unit Price: ");
    System.Console.WriteLine(unitPrice);
  }
}
```

۳. main menu را باز کرده و روی گزینه های Window -> DepartmentStore.cs کلیک کنید.

۴. فایل مورد نظر را به صورت زیر تغییر دهید :

```
public class DepartmentStore
{
  static void Main()
  {
    StoreItem si = new StoreItem();
    si.CreateItem();
    si.Describe();
    System.Console.ReadKey();
  }
}
```

اکنون، برنامه را اجرا کنید. نتیجه ی زیر حاصل می گردد.

```
Department Store
Item #:  792475
Item Name: Aramis Gentlemen Collection 3.4oz JHL Custom Blended Cologne Spray
Item Size: 3.4oz
Unit Price: 48.00
```

۶. پنجره ی DOS را بسته و به محیط برنامه نویسی بازگردید.

متدی که مقدار باز می گرداند

پس از این که متدی مقداری را حمل کرد، باید محصول و نتیجه ی آن را در اختیار دیگر متدها و کلاس ها قرار دهد، در این صورت متد دیگر نمی تواند void یا بی مقدار باشد و سرانجام باید مقداری را بازگرداند. برای تعریف متدی که مقدار باز می گرداند، باید نوع بازگشتی (return type) را در سمت چپ اسم متد قرار داد.

مثال

```
class Exercise
{
  double Operate()
  {
  }
}
```

همچنین می توانید علامت سوأل به نوع داده ی مورد نظر خود اضافه کنید. مثال زیر را در نظر بگیرید.

```
class Exercise
{
  double? Operate()
  {
  }
}
```

در بدنه ی متد، می توان چند متغیر تعریف کرد که فقط توسط خود آن متد مورد استفاده قرار می گیرند. متغیری که در بدنه ی متد تعریف می شود، متغیر محلی نامیده می شود. متغیر محلی تنها از داخل متد قابل دسترسی است (نمی توان به آن خارج از کلاس دسترسی پیدا کرد). پس از تعریف متغیر محلی، متد بلافصله به آن دسترسی پیدا می کند و شما می توانید هر کاری که مناسب می دانید با آن انجام دهید. برای مثال می توانید پیش از استفاده، به متغیر محلی مقدار اختصاص دهید.

پس از این که متد وظیفه ی محوله را انجام داد، باید مقداری را ارائه دهد. برای این منظور باید ابتدا کلید واژه ی return را تایپ کرده و به دنبال آن مقداری را که متد باز می گرداند قرار دهید. مقدار بازگردانده شده باید با نوع بازگشتی متد یکی باشد. مثال زیر را در نظر بگیرید.

```
class Exercise
{
  double Operate()
  {
    return 24.55;
```

```
  }
}
```

اگر چه این نیز صحیح است.

```
class Exercise
{
  double? Operate()
  {
    return 24.55;
  }
}
```

متد ممکن است عبارت (expression) بازگرداند، البته به شرط این که عبارت بیان شده مقداری تولید کند که با نوع بازگشتی هم خوانی داشته باشد. توجه خود را به مثال زیر جلب کنید.

```
class Exercise
{
  double? Operate()
  {
    return 24.55 * 4.16;
  }
}
```


بازگرداندن یک مقدار از متد

۱. در فهرست گزینه ی اصلی، روی Window -> StoreItem.cs کلیک کنید.

۲. فایل را به صورت زیر اصلاح کنید.

```
public class StoreItem
{
  public long itemNumber;
  public string itemName;
  public string size;
  public decimal unitPrice;
  public void CreateItem()
  {
    itemNumber = 792475;
    itemName = "Aramis Gentlemen Collection 3.4oz JHL Custom Blended Cologne Spray";
    size = "3.4oz";
    unitPrice = 48.00m;
  }
  public decimal CalculateTotalValue()
  {
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    return unitPrice;
  }
  public void Describe()
  {
    System.Console.WriteLine("Department Store");
    System.Console.Write("Item #:    ");
    System.Console.WriteLine(itemNumber);
    System.Console.Write("Item Name: ");
    System.Console.WriteLine(itemName);
    System.Console.Write("Item Size: ");
    System.Console.WriteLine(size);
    System.Console.Write("Unit Price: ");
    System.Console.WriteLine(unitPrice);
    System.Console.Write("Total Value: ");
    System.Console.WriteLine(CalculateTotalValue());
  }
}
```

۳. برنامه را اجرا کرده تا نتیجه ی عملیات را مشاهده کنید.

۴. پنجره ی DOS را بسته و به محیط برنامه نویسی خود بازگردید.

مقدمه ای بر تابع Main() یک برنامه ی کاربردی

تاکنون با تابع Main() (که هنگام ایجاد برنامه ی کاربردی جدید و با به کار بردن پنجره ی محاوره ی New Project و به صورت پیش فرض تعریف می شود) کار کردیم. در چنین پیاده سازی به صورت پیش فرض تابع Main() از نوع void می باشد. راه دیگری که برای پیاده سازی تابع Main() وجود دارد، این است که متد، از نوع integer باشد. همین قانون برای تمام متدهای نوع .int قابل اجرا است. تابع Main() می تواند هر نوع integer ای بازگرداند (البته به شرط این که integer بازگشتی معتبر باشد).

مثال

```
class Exercise
{
  char? HaveCharacter()
  {
    return 'G';
  }
  static int Main()
  {
    Exercise exo = new Exercise();
    System.Console.Write("Character: ");
    System.Console.WriteLine(exo.HaveCharacter());
    return 244006;
  }
```

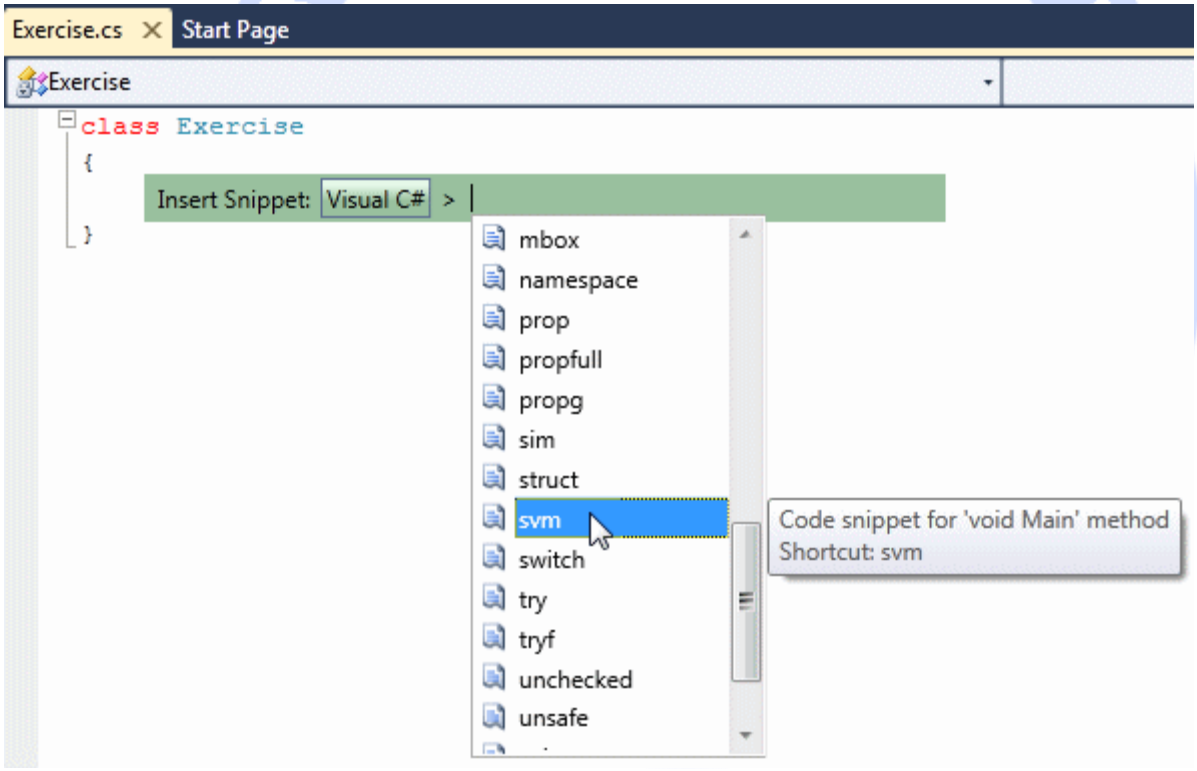
```
}
```

نتیجه ی زیر به دست می آید.

```
Character: G
Press any key to continue...
```

به منظور ایجاد تابع Main با استفاده از skeleton code، روی آن بخشی که می خواهید تابع نام برده را اضافه کنید راست کلیک کرده، سپس گزینه ی Insert Snippet... را انتخاب کنید. حال، روی گزینه ی Visual C# دوبار کلیک کنید.

چنانچه مایلید تابع Main ای ایجاد کنید که مقداری باز نمی گرداند، روی svm دوبار کلیک کنید. (مخفف "static void Main")



به منظور ایجاد تابع Main ای که integer باز می گرداند، روی sim (مخفف"static int Main") دوبار کلیک کنید.

بازگرداندن مقداری از تابع Main()

۱. روی دکمه ی DepartmentStore.cs کلیک کرده تا به آن دسترسی پیدا کنید، سپس آن را به صورت زیر اصلاح کنید :

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
public class DepartmentStore
{
  static int Main()
  {
    StoreItem si = new StoreItem();
    si.CreateItem();
    si.Describe();
    System.Console.ReadKey();
    return 0;
  }
}
```

۲. برنامه را اجرا کرده تا نتیجه ی آن را ببینید.

۳. پنجره ی DOS را بسته و به محیط برنامه نویسی بازگردید.

مبانی آرگومان های متد

متد تکلیفی را انجام می دهد که تکمیل کننده ی عملیات یک کلاس است. متدهایی که در بخش های پیشین به کار بردیم، برای تبادل اطلاعات بین بخش های مختلف برنامه به متغیرهای محلی تکیه می کردند. گاهی اوقات یک متد برای تکمیل عملیات کلاس به چند مقدار نیاز دارد. خصوصیت برجسته ای که مقادیر ذکر شده دارند این است که، متد دیگری که این متد را فرا می خواند باید مقدارهای لازم آن را تامین کند. به مقداری که متد برای تکمیل وظیفه محوله به آن نیاز دارد، آرگومان گفته می شود.

درست مشابه متغیر، یک آرگومان با نوع مقدارش تعریف و نمایش داده می شود. برای مثال، متدی ممکن است به کاراکتر نیاز داشته باشد، در حالی که متدی دیگر به یک رشته نیاز دارد. حتی ممکن است متغیر دیگری به عدد اعشاری (decimal number) نیاز داشته باشد. به عبارت دیگر، متد یا کلاسی که متد دیگری را فرا می خواند، مسئول تامین مقدارهای مناسب و مورد نیاز است (البته یک متد ممکن است خود مکانیزم داخلی برای اعتبار سنجی چنین مقداری داشته باشد).

مقداری که برای متدی فراهم می شود، داخل پرانتزهای متد تایپ می شود. یک متد باید نوع مقداری که به آن نیاز دارد را مشخص کند، به همین خاطر آرگومان مورد نظر با نوع داده ی معین و یک اسم نمایش داده می شود.

تصور کنید که می خواهیم متدی تعریف کنیم که طول ضلع یک مربع را نمایش دهد. چنین متدی به صورت زیر تعریف می شود.

```
class Exercise
{
  void ShowCharacter(char c)
```

```
  {
  }
}
```

در صورتی که نوع داده ی آرگومان از نوع اولیه بود، برای این که نشان دهیم نوع داده ی مزبور می تواند مقدار تهی (null value) داشته باشد، به آن علامت سوأل اضافه می کنیم. نمونه ی آن را در زیر مشاهده می کنید :

```
class Exercise
{
  void ShowCharacter(char? c)
  {
  }
}
```

می توان در بدنه ی یک متد مقدار آرگومان را لحاظ کرد (هم چنین می توان مقدار آرگومان را در بدنه ی متد به کار نبرد). مثال زیر مقدار آرگومان را در بدنه ی متد نمایش می دهد.

```
class Exercise
{
  void ShowCharacter(char c)
  {
    System.Console.WriteLine(c);
  }
}
```

نمونه ی زیر نیز معتبر و صحیح است.

```
class Exercise
{
  void ShowCharacter(char? c)
  {
    System.Console.WriteLine(c);
  }
}
```

به همین شیوه می توان مقدار ارائه شده را به صورت دلخواه تنظیم و اداره کرد.

فراخوانی متدی که آرگومان می گیرد

در وهله ی اول برای استفاده از یک متد باید آن را فرا خواند. به منظور فراخوانی متدی که آرگومان می گیرد، ابتدا باید مقداری برای آرگومان فراهم کنید، در غیر این صورت پیغام خطا (error) دریافت می کنید. همچنین باید مقدار مناسب را در اختیار آرگومان قرار دهیم، در غیر این صورت متد آن گونه که باید کار خود را درست انجام نمی دهد و نتیجه ی اتکاناپذیر و نامعتبر به دست می دهد.

چنانچه متد حین فراخوانی، آرگومان بگیرد، می توانید داخل پرانتز آن مقداری تایپ کنید.

مثال

```
class Exercise
{
  void ShowCharacter(char c)
  {
    System.Console.WriteLine(c);
  }
  static int Main()
  {
    Exercise exo = new Exercise();
    System.Console.Write("Character: ");
    exo.ShowCharacter('W');
    return 0;
  }
}
```

متدی که آرگومان می گیرد، همچنین می تواند متغیرهای محلی خود را تعریف کند.

ارسال آرگومان ها

۱. روی دکمه ی StoreItem.cs کلیک کرده تا فایل مورد نظر باز شود، سپس آن را به صورت زیر اصلاح کنید.

```
public class StoreItem
{
  public long itemNumber;
  public string itemName;
  public string size;
  public decimal unitPrice;
  public void CreateItem()
  {
    itemNumber = 911792;
    itemName = "Girls Tillie Poplin Dress";
    size = "12";
    unitPrice = 110.00m;
  }
  public decimal CalculateTotalValue(int qty)
  {
    return unitPrice * qty;
  }
  public void Describe()
  {
    int quantity = 6;
    System.Console.WriteLine("Department Store");
    System.Console.Write("Item #:    ");
    System.Console.WriteLine(itemNumber);
    System.Console.Write("Item Name:  ");
    System.Console.WriteLine(itemName);
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    System.Console.Write("Item Size: ");
    System.Console.WriteLine(size);
    System.Console.Write("Unit Price: ");
    System.Console.WriteLine(unitPrice);
    System.Console.Write("Quantity: ");
    System.Console.WriteLine(quantity);
    System.Console.Write("Total Value: ");
    System.Console.WriteLine(CalculateTotalValue(quantity));
  }
}
```

۲. اکنون برنامه را اجرا کنید. نتیجه ی زیر حاصل می گردد.

```
Department Store
Item #:  911792
Item Name: Petite Tropical Wool Pencil Skirt
Item Size: 12
Unit Price: 110.00
Quantity: 6
Total Value: 660.00
```

۳. پنجره ی DOS را بسته و به محیط برنامه نویسی بازگردید.

ارسال چندین آرگومان به یک متد

یک متد می تواند آرگومان های متعدد داشته باشد. البته، برای تک تک آرگومان ها باید یک اسم و نوع داده در نظر گرفته شود. آرگومان ها با ویرگول (،) از هم جدا می شوند. به مثال زیر توجه کنید.

```
class Exercise
{
  void ShowEmployee(long employeeNumber, string fullName,
        char gender, double hourlySalary)
  {
  }
}
```

در صورت الزام، به آرگومان هایی که از نوع اولیه ی داده هستند، علامت سوال اضافه می کنیم.

توجه

از این بخش درس به بعد، به منظور اشاره به متدی که متعلق به یک کلاس هست از قرداد زیر استفاده می کنیم.

```
ClassName.MethodName(Arguments)
```

توجه داشته باشید که آیتم داخل پرانتز به آرگومانی که متد می گیرد اشاره دارد.

توجه

از این بخش درس به بعد، می نویسیم " دستور نحوی این متد هست ".

```
ReturnValue ClassName.MethodName(Arguments);
```

با این کار داریم نوع بازگشتی (return type)، اسم مورد نظر، و تعداد آرگومان های یک متد را ارائه می دهیم.

در قرارداد جدید ما، دستور نحوی را با نقطه ویرگول (;) خاتمه می دهیم. بر خلاف زبان های C، C++، Object Pascal، Visual Basic، زبان برنامه نویسی C# و Java این اجازه را به شما نمی دهد که اول تابعی را تعریف کنید، سپس آن را در بخش دیگر فایل یا فایلی مجزا پیاده سازی کنید. ما از نقطه ویرگول برای نشان دادن ساختار یک متد استفاده می کنیم (و نه برای خود تعریف).

هنگام تعریف متغیر، می توانید ۰، ۱ یا حتی تمامی آرگومان ها را هر طور که مایلید به کار ببرید. برای مثال، می توانید مقادیر آن را با کنسول نشان دهید. به مثال های زیر توجه کنید.

```
 class Exercise
{
  void ShowEmployee(long employeeNumber, string fullName,char gender, double hourlySalary)
  {
    System.Console.Write("Employee #:  ");
    System.Console.WriteLine(employeeNumber);
    System.Console.Write("Full Name:  ");
    System.Console.WriteLine(fullName);
    System.Console.Write("Gender:  ");
    System.Console.WriteLine(gender);
    System.Console.Write("Hourly Salary: ");
    System.Console.WriteLine(hourlySalary);
    System.Console.ReadKey();
  }
}
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

راه های مختلفی برای فراخوانی متدی که آرگومان می گیرد وجود دارد. دیدیم که هنگام فراخوانی متدی که آرگومان می گیرد، می توان مقداری در پرانتزهای متد تایپ کرد. به همین ترتیب هم، هنگام فراخوانی متدی که چندین آرگومان می گیرد، کافی است مقدار مربوط را در آن آرگومان تایپ کنید. مثال های آن را زیر مشاهده می کنید.placeholder

```
class Exercise
{
  void ShowEmployee(long employeeNumber, string fullName,
            char gender, double hourlySalary)
  {
    System.Console.WriteLine("Employee Record");
    System.Console.WriteLine("------------------------------");
    System.Console.Write("Employee #:  ");
    System.Console.WriteLine(employeeNumber);
    System.Console.Write("Full Name:  ");
    System.Console.WriteLine(fullName);
    System.Console.Write("Gender:  ");
    System.Console.WriteLine(gender);
    System.Console.Write("Hourly Salary: ");
    System.Console.WriteLine(hourlySalary);
  }
  static int Main()
  {
    Exercise exo = new Exercise();
    exo.ShowEmployee(572948, "Andrew Bridges", 'M', 22.85D);
    return 0;
  }
}
```

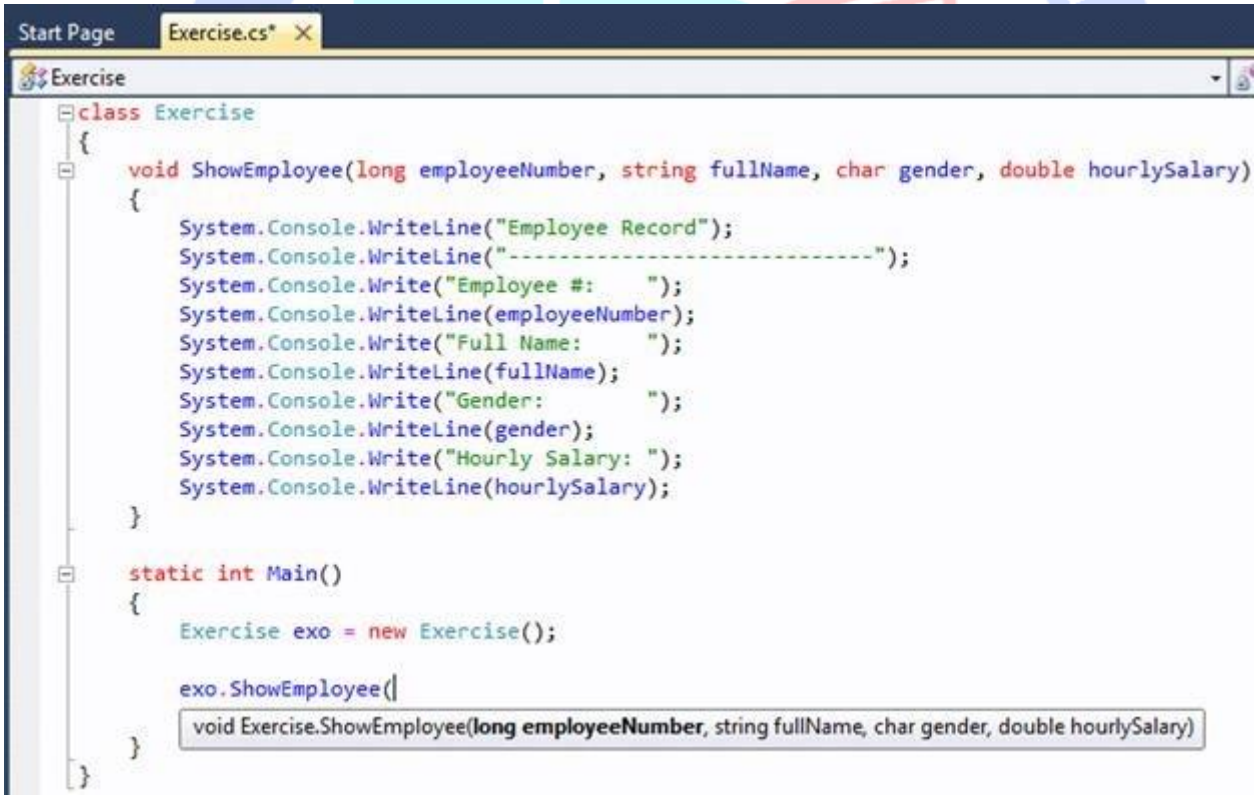
این نتیجه را به دست می دهد.

```
Employee Record
------------------------------
Employee #:  572948
Full Name:  Andrew Bridges
Gender:  M
Hourly Salary: 22.85
Press any key to continue...
```

فراخوانی آرگومان با استفاده از اسم

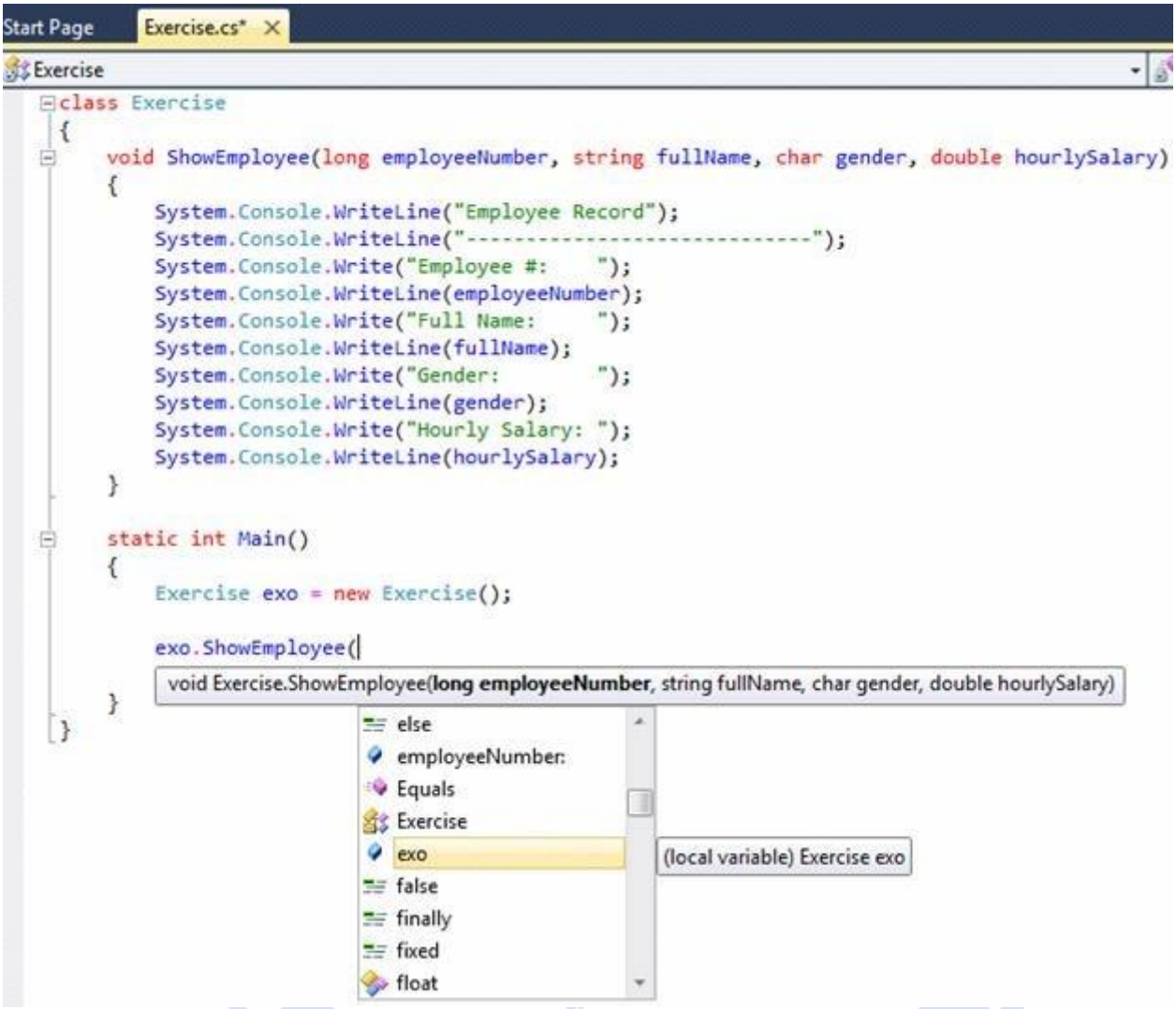
چنانچه متدی فراخوانی کنید که آرگومان های متعددی می گیرد، نیازی به استفاده از placeholder دقیق هر آرگومان نیست. در عوض می توان به هر آرگومان با اسم آن اشاره کرد. برای این منظور، در پرانتزهای متد، ابتدا اسم آرگومان را تایپ کنید و به دنبال آن علامت (:) و مقدار مناسب را درج کنید.

در صورت کار با برنامه ی Microsoft Visual Studio، پس از وارد کردن اسم متد و "(" ، قاعدتاً باید بتوانید فهرستی از اسم های آرگومان ها را به صورت زیر مشاهده کنید.



سپس، می توانید اسم آرگومان را تایپ کنید. می توانید هم به جای تایپ کردن آن، کلید های Ctrl + Space را فشار دهید. حال، فهرستی ظاهر می شود. در فهرست مذکور اسم های آرگومان ها قابل مشاهده می باشد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



اکنون می توانید اسم آرگومان را از فهرست انتخاب کنید. همان طور که پیش تر ذکر شد، پس از اسم آرگومان، علامت " : " و به دنبال آن مقدار آرگومان را وارد می کنیم. به همین ترتیب، می توان به هر آرگومان با استفاده از اسم دسترسی پیدا کرد، سپس مقدار به آن تخصیص داد و آن ها را با ویرگول (،) از هم جدا کرد. مثال :

```
class Exercise
{
  void ShowEmployee(long employeeNumber, string fullName,char gender, double hourlySalary)
  {
    System.Console.WriteLine("Employee Record");
    System.Console.WriteLine("----------");
    System.Console.Write("Employee #:  ");
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    System.Console.WriteLine(employeeNumber);
    System.Console.Write("Full Name:    ");
    System.Console.WriteLine(fullName);
    System.Console.Write("Gender:      ");
    System.Console.WriteLine(gender);
    System.Console.Write("Hourly Salary: ");
    System.Console.WriteLine(hourlySalary);
  }
  static int Main()
  {
    Exercise exo = new Exercise();
    exo.ShowEmployee(fullName: "Annette Greens", hourlySalary: 16.85,employeeNumber: 519427, gender: 'F');
    return 0;
  }
}
```

نتیجه

```
Employee Record
------------------------------
Employee #:  519427
Full Name:    Annette Greens
Gender:      F
Hourly Salary: 16.85
Press any key to continue...
```

ارسال آرگومان ها به متد یک کلاس

روش های ارسال آرگومان

ارسال آرگومان با استفاده از مقدار

هنگام فراخوانی متدی که چندین آرگومان می گیرد، گفتیم که ابتدا باید مقدارهای مورد نیاز آرگومان ها را فراهم کنیم. این به آن خاطر است که (وجود) آرگومان الزامی بوده و متد فراخوان (calling method) باید (هنگام فراخوانی متدی که چندین آرگومان دارد) مقداری معتبر ارائه دهد. به این روش از ارائه ی مقدار برای آرگومان، روش ارسال آرگومان با استفاده از مقدار گفته می شود.

نحوه ی ارسال آرگومان با استفاده از مقدار

۱. Microsoft Visual Studio را اجرا کنید.

۲. برای ایجاد برنامه ی کاربردی جدید، به فهرست اصلی مراجعه کرده و گزینه های File -> New Project... را کلیک کنید.

۳. Empty Project را از لیست میانی انتخاب کنید.

۴. اسم را به NationalBank1 تغییر داده، سپس کلید Enter را فشار دهید.

۵. به منظور ایجاد فایل جدید، در Solution Explorer، راست کلیک کرده و طبق دستور زیر به پیش بروید : NationalBank1 -> Add -> New Item...

۶. در لیست میانی روی Code File کلیک کنید.

۷. اسم فایل را به Management تغییر دهید.

۸. روی گزینه ی Add کلیک کنید.

۹. در فایل خالی Management.cs، دستورات زیر وارد کنید.

```
public class Management
{
  private void ShowAccountInformation(string acntNbr, string name, short PIN, double balance)
  {
    System.Console.WriteLine("================================");
    System.Console.WriteLine("Account Information");
    System.Console.WriteLine("----------------------------");
    System.Console.Write("Account #:  ");
    System.Console.WriteLine(acntNbr);
    System.Console.Write("Customer Name: ");
    System.Console.WriteLine(name);
    System.Console.Write("PIN:      ");
    System.Console.WriteLine(PIN);
    System.Console.Write("Balanace:  ");
    System.Console.WriteLine(balance);
    System.Console.WriteLine("================================");
  }
  public static int Main()
  {
    Management man = new Management();
    string accountNumber = "248-050842-749";
    string customerName = "Ann Kelley";
    short pin = 8648;
    double currentBalance = 350.00D;
    man.ShowAccountInformation(accountNumber, customerName,pin, currentBalance);
    System.Console.ReadKey();
    return 0;
  }
}
```

۱۰. برای اجرای دستورات، کلید F5 را فشار دهید.

```
==============================
Account Information
------------------------------
Account #:  248-050842-749
Customer Name: Ann Kelley
PIN:  8648
Balanace:  350
==============================
```

۱۱. پس از مشاهده ی نتایج، با فشار دادن کلید Enter از پنجره ی DOS خارج شوید.

ارسال آرگومان با استفاده از ارجاع

برنامه ی زیر را در نظر بگیرید.

```
public class Exercise
{
  void GetDeposit(decimal amount)
  {
    amount = 450;
    System.Console.WriteLine("In GetDeposit()");
    System.Console.Write("Amouont: $");
    System.Console.WriteLine(amount);
  }
  static int Main()
  {
    decimal amt = 0;
    var exo = new Exercise();
    System.Console.WriteLine("In Main()");
    System.Console.Write("Amouont: $");
    System.Console.WriteLine(amt);
    System.Console.WriteLine("-------------------------");
    exo.GetDeposit(amt);
    System.Console.WriteLine("-------------------------");
    System.Console.WriteLine("Bank in Main()");
    System.Console.Write("Amouont: $");
    System.Console.WriteLine(amt);
    return 0;
  }
}
```

نتیجه ی زیر حاصل می گردد.

```
In Main()
Amouont: $0
-------------------------
In GetDeposit()
Amouont: $450
-------------------------
Bank in Main()
Amouont: $0
Press any key to continue...
```

همان طور که مشاهده می کنید، مقدار آرگومان قبل و پس از فراخوانی متد GetDeposit() یکسان است و هیچ فرقی نمی کند.

هنگامی که متغیری را در برنامه ی خود تعریف می کنید، compiler مقداری حافظه به آن تخصیص می دهد. در مورد متغیر دو چیز از اهمیت بالایی برخوردار است : مقدار متغیر و محل قرار گیری آن در حافظه. به مکان یا محل قرارگیری متغیر در حافظه، آدرس متغیر می گویند.

اگر آرگومان را با استفاده از اسم فراخوانی کنید، compiler تنها کپی ای از مقدار آرگومان تهیه کرده، سپس آن را به متد فراخوان (calling method) می دهد. اگرچه متد فراخوان مقدار آرگوامان ذکر شده را دریافت می کند و می تواند آن را به هر شکل ممکن به کار برد، اما نمی تواند آن را (برای همیشه) تغییر دهد. در صورت نیاز می توان با فراخوانی متد، مقدار یک آرگومان را تغییر داد (modify). اگر می خواهید متد فراخوان مقدار آرگومان ارائه شده را تغییر داده، سپس مقدار تغییر داده شده (modified) را بازگرداند، باید ارگومان را با استفاده از ارجاع (reference) ارسال کنید.

حال، برای ارسال آرگومان با استفاده از ارجاع، حین تعریف متد، باید کلیدواژه ی ref را پیش از نوع داده ی آرگومان تایپ کرد.

مثال

```
public class Exercise
{
  void GetDeposit(ref decimal amount)
  {
  }
}
```

در متد، می توانید آرگومان را به کار ببرید (یا آن را نادیده بگیرید و مورد استفاده قرار ندهید). هنگام فراخوانی متد، اسم آرگومان را پس از کلیدواژه ی ref به کار ببرید. به مثال زیر توجه کنید.

```
public class Exercise
{
```

```
    void GetDeposit(ref decimal amount)
    {
    }
    static int Main()
    {
      decimal amt = 0;
      var exo = new Exercise();
      System.Console.WriteLine("In Main()");
      System.Console.Write("Amount: $");
      System.Console.WriteLine(amt);
      exo.GetDeposit(ref amt);
      System.Console.WriteLine("Bank in Main()");
      System.Console.Write("Amount: $");
      System.Console.WriteLine(amt);
      return 0;
    }
}
```

نتیجه

```
In Main()
Amount: $0
Bank in Main()
Amount: $0
Press any key to continue...
```

دلیل اصلی ارسال آرگومان به وسیله ی ارجاع، این است که به متد اجازه ی تغییر مقدار آرگومان داده شود. برای این منظور، لازم است مقدار آرگومان را در متد تغییر داد.

مثال

```
public class Exercise
 {
    void GetDeposit(ref decimal amount)
    {
      amount = 450;
      System.Console.WriteLine("In GetDeposit()");
      System.Console.Write("Amount: $");
      System.Console.WriteLine(amount);
    }
    static int Main()
    {
      decimal amt = 0;
      var exo = new Exercise();
```

```
    System.Console.WriteLine("In Main()");
    System.Console.Write("Amount: $");
    System.Console.WriteLine(amt);
    System.Console.WriteLine("-------------------------");
    exo.GetDeposit(ref amt);
    System.Console.WriteLine("-------------------------");
    System.Console.WriteLine("Bank in Main()");
    System.Console.Write("Amount: $");
    System.Console.WriteLine(amt);
    return 0;
  }
}
```

نتیجه

```
In Main()
Amount: $0
-------------------------
In GetDeposit()
Amount: $450
-------------------------
Bank in Main()
Amount: $450
Press any key to continue...
```

همان طور که در مثال بالا مشاهده کردید، این بار مقدار متغیر پس از فراخوانی تابع تغییر یافته.

می توانید کلیه ی آرگومان ها را به عنوان ارجاع ارسال کنید. این که کدام آرگومان با مقدار ارسال شود و کدام با ارجاع بستگی به این دارد که شما می خواهید متد فراخوانده آرگومان را تغییر دهد یا نه.

دیدیم که (یک) تابع می تواند تنها یک مقدار بازگرداند، زیرا تنها یک کلیدواژه ی return وجود دارد. خوشبختانه، ارجاع (reference) با داشتن قابلیت ارسال چندین آرگومان، به متد اجازه می دهد چندین مقدار بازگرداند.

نحوه ی ارسال آرگومان با استفاده از ارجاع

۱. فایل Management.cs را به صورت زیر اصلاح کنید.

```
public class Management
{
  private void ShowAccountInformation(string acntNbr, string name, short PIN, double balance)
  {
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    System.Console.WriteLine("================================");
    System.Console.WriteLine("Account Information");
    System.Console.WriteLine("----------------------------");
    System.Console.Write("Account #:  ");
    System.Console.WriteLine(acntNbr);
    System.Console.Write("Customer Name:");
    System.Console.WriteLine(name);
    System.Console.Write("PIN:  ");
    System.Console.WriteLine(PIN);
    System.Console.Write("Balanace:  ");
    System.Console.WriteLine(balance);
    System.Console.WriteLine("================================");
  }

  private double GetDeposit(ref double amount)
  {
    double deposit = 225.55;
    amount = amount + deposit;
    return deposit;
  }
  private double GetWithdrawal(ref double amount)
  {
    double withdrawal = 265.25d;
    amount = amount - withdrawal;
    return withdrawal;
  }
  public static int Main()
  {
    Management man = new Management();
    string accountNumber = "248-050842-749";
    string customerName = "Ann Kelley";
    short pin = 8648;
    double currentBalance = 350.00D;
    double? depositAmount = null;
    double? withdrawalAmount = null;
    man.ShowAccountInformation(accountNumber, customerName,pin, currentBalance);
    depositAmount = man.GetDeposit(ref currentBalance);
    System.Console.WriteLine("After a new deposit");
    man.ShowAccountInformation(accountNumber, customerName,pin, currentBalance);
    withdrawalAmount = man.GetWithdrawal(ref currentBalance);
    System.Console.WriteLine("After a withdrwal");
    man.ShowAccountInformation(accountNumber, customerName,pin, currentBalance);
    System.Console.ReadKey();
    return 0;
  }
}
```

۲. کلید F5 را زده تا برنامه اجرا شود.

```
==============================
Account Information
------------------------------
Account #:  248-050842-749
Customer Name: Ann Kelley
PIN:  8648
Balanace:  350
==============================
After a new deposit
==============================
Account Information
------------------------------
Account #:  248-050842-749
Customer Name: Ann Kelley
PIN:  8648
Balanace:  575.55
==============================
After a withdrwal
==============================
Account Information
------------------------------
Account #:  248-050842-749
Customer Name: Ann Kelley
PIN:  8648
Balanace:  310.3
==============================
```

۳. Enter را زده تا پنجره ی DOS بسته شود.

ارسال آرگومان out

روش دیگر ارسال آرگومان با استفاده از کلیدواژه ی out می باشد. به منظور ارسال آرگومان out، داخل پرانتز متد پیش از نوع داده ی آرگومان کلیدواژه ی out را تایپ می کنیم.

مثال

```
public class Exercise
{
  void GetWithdrawal(out decimal amount)
  {
  }
```

```
}
```

تفاوت اصلی بین کلید واژه های ref و out این است که در صورت ارسال آرگومان با کلید واژه ی out، آرگومان باید حتماً داخل خود متد مقداردهی اولیه شود.

مثال

```
public class Exercise
{
  void GetWithdrawal(out decimal amount)
  {
    amount = 265.00m;
  }
}
```

هنگام فراخوانی متدی که آرگومان out می گیرد، باید کلید واژه ی out را پیش از آرگومان استفاده کنید. به مثال زیر توجه کنید.

```
public class Exercise
  {
    void GetDeposit(ref decimal amount)
    {
      amount = 450.00M;
      System.Console.WriteLine("In GetDeposit()");
      System.Console.Write("Amount: $");
      System.Console.WriteLine(amount);
    }
    void GetWithdrawal(out decimal amount)
    {
      amount = 265.00m;
      System.Console.WriteLine("In GetWithdrawal()");
      System.Console.Write("Amount: $");
      System.Console.WriteLine(amount);
    }
    static int Main()
    {
      decimal depositAmount = 0M;
      decimal withdrawalAmount = 0M;
      var exo = new Exercise();
      System.Console.WriteLine("In Main()");
      System.Console.Write("Amount: $");
      System.Console.WriteLine(depositAmount);
      System.Console.WriteLine("-------------------------");
      exo.GetDeposit(ref depositAmount);
      System.Console.WriteLine("-------------------------");
      System.Console.WriteLine("Bank in Main()");
      System.Console.Write("Amount: $");
      System.Console.WriteLine(depositAmount);
      System.Console.WriteLine("-------------------------");
      exo.GetWithdrawal(out withdrawalAmount);
```

```
    System.Console.WriteLine("-------------------------");
    System.Console.WriteLine("Bank in Main()");
    System.Console.Write("Amount: $");
    System.Console.WriteLine(withdrawalAmount);
    return 0;
  }
}
```

چنانچه آرگومانی را با کلید واژه ی out ارسال کنید، تمام تغییرات وارد آمده بر آرگومان حتی پس از اتمام متد، نگه داشته می شوند.

```
In Main()
Amount: $0
-------------------------
In GetDeposit()
Amount: $450.00
-------------------------
Bank in Main()
Amount: $450.00
-------------------------
In GetWithdrawal()
Amount: $265.00
-------------------------
Bank in Main()
Amount: $265.00
Press any key to continue...
```

سربارگذاری متد (method overloading)

تابع زیر به نام CalculateArea را در نظر بگیرید.

```
public class Geometry
{
  double CalculateArea()
  {
    return 0D;
  }
}
```

هنگامی که متدی در کلاس ایجاد کرده و پروژه ای می سازید، compiler لیستی از آن متدها تهیه می کند. به لیست مزبور virtual table (جدول مجازی) می گویند. در جدول بیان شده، متد با استفاده از اسمش کد گذاری می شود.

Method Name	Encoding

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

CalculateArea	Calculate@None

در صورتی که متد آرگومانی داشت، جدول بالا نوع داده ی آرگومان را نمایش می دهد. نوع بازگشتی متد و اسم های آرگومان ها هیچ کدام مهم نیستند. در صورتی که کلاس دربردارنده ی متدهایی با اسامی متفاوتی بود، متدها به راحتی از هم قابل تشخیص خواهند بود. گاهی اوقات لازم است متدهای مختلفی داشته باشید که همگی یک کار مشابه را انجام می دهند ولی با تفاوت های جزئی (این دیگر بستگی به شما دارد که این تفاوت ها چه باشند). یکی از گزینه های پیش رو این است که اسم متدها را یکی انتخاب کنید. به قابلیت داشتن چندین متد با اسم مشابه در یک کلاس method overloading می گویند.

برای اجرای method overloading، هنگام ایجاد virtual table، تک تک درایه ها (entry) در لیست نام برده باید منحصر به فرد باشند. همان طور که پیش تر ذکر شد، نوع بازگشتی متد داخل جدول قرار داده نمی شود، در نتیجه نمی توان تفاوت ها را بر آن اساس پایه گذاری کرد. اگر سعی کنید دو متد ایجاد کنید که هیچ یک آرگومان نمی گیرند، طبیعتاً تشخیص آن دو متد از یک دیگر ممکن نخواهد بود. راه حل مسئله فوق این است که حداقل یکی از آن دو متد آرگومان بگیرند. به مثال زیر توجه کنید.

```
public class Geometry
{
  // Unknown Shape
  public double CalculateArea()
  {
    return 0D;
  }
  // Square
  public double CalculateArea(double side)
  {
    return side * side;
  }
}
```

Method Name	Argument(s)	Encoding
CalculateArea		Calculate@None
CalculateArea	Double	Calculate@Double

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

همان طور که انتظار می رفت، این بار هر متد رمزگذاری (encoding) جداگانه ای (متفاوتی) دارد. مثال زیر کد بالا را به کار می برد.

```
public class Geometry
{
  // Unknown Shape
  public double CalculateArea()
  {
    return 0D;
  }
  // Square
  public double CalculateArea(double side)
  {
    return side * side;
  }
}
public class Exercise
{
  static int Main()
  {
    Geometry geo = new Geometry();
    System.Console.WriteLine("Geometric Shapes");
    System.Console.WriteLine("Calculation of Areas");
    System.Console.Write("Square: ");
    System.Console.WriteLine(geo.CalculateArea(26.38));
    return 0;
  }
}
```

نتیجه ی زیر به دست می آید.

```
Geometric Shapes
Calculation of Areas
Square: 695.9044
Press any key to continue...
```

به خاطر داشته باشید که اسم یک آرگومان بخشی از رمزگذاری (encoding) آن در virtual table محسوب نمی شود. بنابراین، نمی توان دو متد به کاربرد که هر دو یک آرگومان یکسان داشته باشد. کد زیر را در نظر بگیرید.

```
public class Geometry
{
  // Unknown Shape
  public double CalculateArea()
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    {
        return 0D;
    }
    // Square
    public double CalculateArea(double side)
    {
        return side * side;
    }
    // Circle
    public double CalculateArea(double radius)
    {
        return radius * radius * 3.14159;
    }
}
```

عملیات بالا نتیجه ی زیر را به دست می دهد.

Method Name	Argument(s)	Encoding
CalculateArea		Calculate@None
CalculateArea	Double	Calculate@Double
CalculateArea	Double	Calculate@Double

در نتیجه، زمانی که پروژه را به وجود می آورید، compiler پیغام خطای زیر را تولید می کند.

```
Error 1 Type 'Geometry' already defines a member called 'CalculateArea'
with the same parameter types
C:\Exercise1\Exercise.cs  16  19  Overloading
```

یکی از قواعد اضافه بار گذاری متد این است که متد گفته شده می تواند آرگومان هایی از نوع مختلف داشته باشد. به مثال زیر توجه کنید.

```
public class Geometry
{
    // Unknown Shape
    public double CalculateArea()
    {
        return 0D;
    }
    // Square
```

```
    public double CalculateArea(double side)
    {
      return side * side;
    }
    // Circle
    public double CalculateArea(double radius)
    {
      return radius * radius * 3.14159;
    }
}
```

در بیشتر موارد، شما به دلایلی نمی خواهید آرگومان متفاوتی (از نوع دیگر) ارسال کنید. گفتیم که نوع داده ی آرگومان در داخل virtual table قرار می گیرد. اما اگر هر متد ، شمار متفاوتی آرگومان بگیرد چه می شود؟ مثال زیر را در نظر بگیرید.

```
public class Geometry
{
    // Unknown Shape
    public double CalculateArea()
    {
      return 0D;
    }
    // Square
    public double CalculateArea(double side)
    {
      return side * side;
    }
    // Rectangle
    public double CalculateArea(double length, double height)
    {
      return length * height;
    }
}
```

نتیجه ی زیر را به دست می دهد.

Method Name	Argument(s)	Encoding
CalculateArea		Calculate@None
CalculateArea	double	Calculate@Double

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

CalculateArea	double، double	Calculate@Double@Double

همان طور که در جدول بالا مشاهده می کنید، رمزگذاری (encoding) متد دوم و سوم متفاوت هستند. نمونه ای از فراخوانی متدها را در مثال زیر مشاهده می کنید.

```
public class Geometry
{
  // Unknown Shape
  public double CalculateArea()
  {
    return 0D;
  }
  // Square
  public double CalculateArea(double side)
  {
    return side * side;
  }
  // Rectangle
  public double CalculateArea(double length, double height)
  {
    return length * height;
  }
}
public class Exercise
{
  static int Main()
  {
    var geo = new Geometry();
    System.Console.WriteLine("Geometric Shapes");
    System.Console.WriteLine("Calculation of Areas");
    System.Console.Write("Square: ");
    System.Console.WriteLine(geo.CalculateArea(26.38));
    System.Console.Write("Rectangle: ");
    System.Console.WriteLine(geo.CalculateArea(39.17, 26.38));
    System.Console.ReadKey();
    return 0;
  }
}
```

نتیجه

```
eometric Shapes
alculation of Areas
```

```
quare:  695.9044
ectangle: 1033.3046
ress any key to continue...
```

اگرچه در مثال بالا از دو آرگومان نوع دابل (double argument) برای متد سوم استفاده کردیم، آرگومان می تواند از هر نوعی باشد. این تصمیم دیگر با شما است که چی متد را به وجود آورد. بنابراین، قاعده ی دیگر سربارگذاری متد ایجاب می کند هر متد آرگومان های متفاوتی داشته باشد.

یکی از خصوصیات بیشتر زبان های برنامه نویسی کامپیوتر از جمله زبان های مبنای C، C++، Java، C# و دیگر زبان ها مثل : Pascal، Visual Basic این است که برنامه نویس نیازی به استفاده از آرگومان داخل متد ندارد. از این امکان می توان برای ایجاد دو نسخه ی کاملاً متفاوت متد استفاده کرد که هر دو عملکرد (behavior) مشابه دارند : می توان آرگومانی ارسال کرد که در آینده مورد استفاده قرار نمی گیرد ولی منجر به ایجاد درایه ی جدید و متمایز در virtual table می شود. کد زیر را در نظر بگیرید.

```
public class Geometry
{
  // Unknown Shape
  public double CalculateArea()
  {
    return 0D;
  }
  // Square
  public double CalculateArea(double side)
  {
    return side * side;
  }
  // Rectangle
  public double CalculateArea(double length, double height)
  {
    return length * height;
  }
  // Circle
  public double CalculateArea(double radius, int unused)
  {
    return radius * radius * 3.14159;
  }
}
```

نتیجه ی زیر حاصل می گردد.

Method Name	Argument(s)	Encoding

CalculateArea		Calculate@None
CalculateArea	double	Calculate@Double
CalculateArea	double، double	Calculate@Double@Double
CalculateArea	double، double	Calculate@Double@Integer

همان طور که مشاهده می کنید رمز گذاری ها متمایز هستند.

به خاطر دارید که برای محاسبه ی محیط های یک مربع یا دایره، به تنها یک مقدار (value) نیاز دارید. به منظور تمیز دادن / وجه تمایز قائل شدن بین محاسبات این دو محیط در کد، آرگومانی متفاوت (دومی) به محیط دایره ارسال می کردیم. ما در مثال خود آرگومان را "unused" نامیدیم، ولی شما می توانید هر اسمی که دوست دارید برای آن انتخاب کنید (یکی از تفاوت های بین زبان های C++ و C# این است که در C++ می توان اسم آرگومان را حذف کرد. بنابراین، اگر کد فوق را با زبان C++ می نوشتیم دیگر نیازی به نام گذاری آرگومان وجود نداشت).

هنگام فراخوانی متدی که دو آرگومان می گیرد، توجه داشته باشید که حتماً باید مقداری به آرگومان دومی ارسال شود. به این خاطر که متد آرگومان گفته شده را به کار نمی برد، فرقی نمی کند چه مقداری به آن ارسال کنید، مادام این که با نوع متد هم خوانی و مطابقت داشته باشد.

مثال

```
public class Geometry
{
  // Unknown Shape
  public double CalculateArea()
  {
    return 0D;
  }
  // Square
  public double CalculateArea(double side)
  {
    return side * side;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

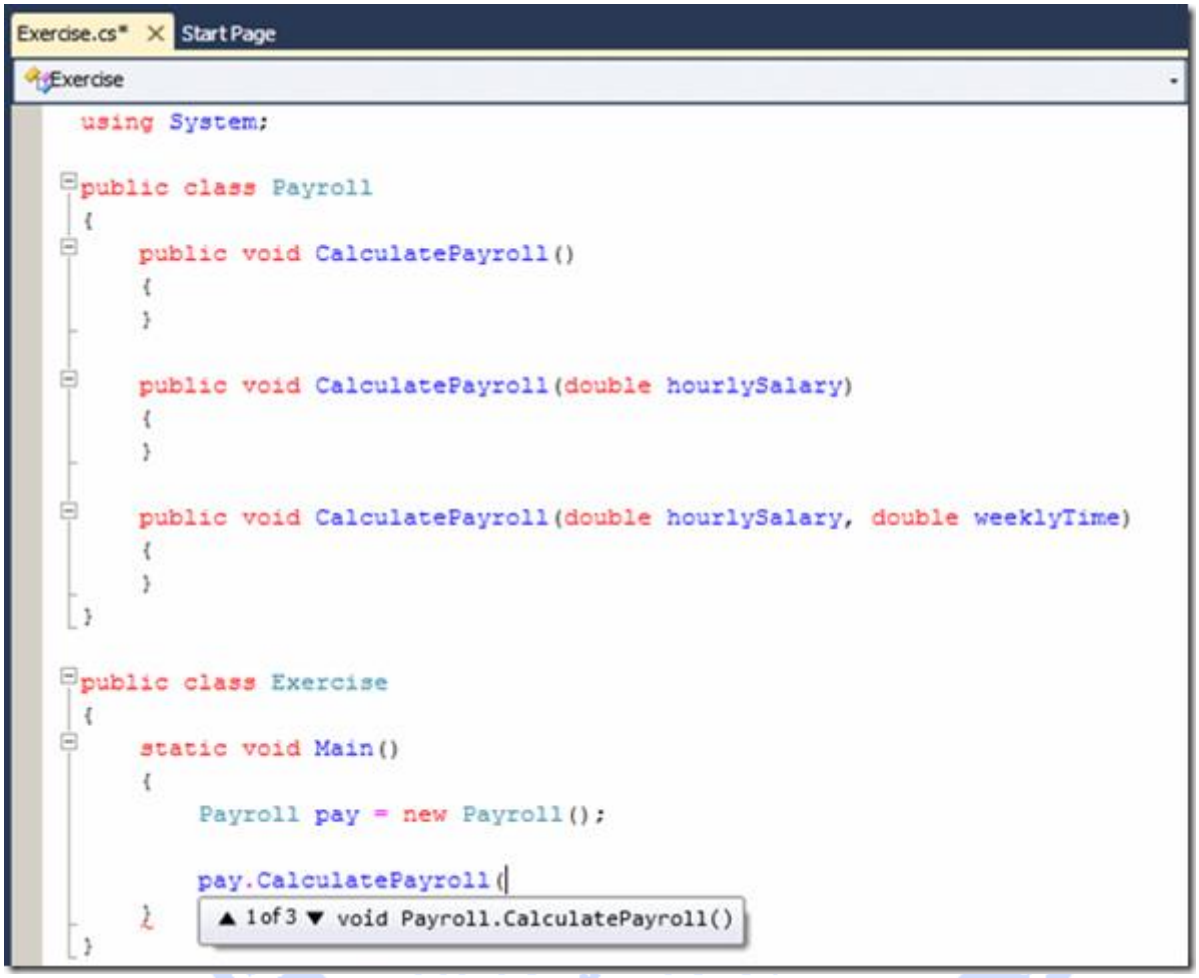
```
    }
    // Rectangle
    public double CalculateArea(double length, double height)
    {
      return length * height;
    }
    // Circle
    public double CalculateArea(double radius, int unused)
    {
      return radius * radius * 3.14159;
    }
}
public class Exercise
{
    static int Main()
    {
      var geo = new Geometry();
      System.Console.WriteLine("Geometric Shapes");
      System.Console.WriteLine("Calculation of Areas");
      System.Console.Write("Square: ");
      System.Console.WriteLine(geo.CalculateArea(26.38));
      System.Console.Write("Rectangle:");
      System.Console.WriteLine(geo.CalculateArea(39.17, 26.38));
      System.Console.Write("Circle: ");
      System.Console.WriteLine(geo.CalculateArea(26.38, 0));
      System.Console.ReadKey();
      return 0;
    }
}
```

نتیجه

```
Geometric Shapes
Calculation of Areas
Square: 695.9044
Rectangle:1033.3046
Circle: 2186.246303996
Press any key to continue...
```

سربارگذاری متد در .NET Framework کاربرد فراوانی دارد. بنابراین، لازم است قواعد آن را هنگام استفاده از این برنامه به خاطر داشته باشید. خوشبختانه این قواعد بسیار ساده هستند : متدها یا باید شمار متفاوت / تعداد مختلفی آرگومان داشته باشند و یا آرگومان های آن ها از نوع متفاوت باشند (نوع آرگومان های ارسال شده به آن ها از هم متمایز باشند).

هنگامی که متدی overload می شود، در code editor، IntelliSense برچسبی ۱ از x نمایش می دهد. x نشانگر تعداد نسخه های متد می باشد. در زیر تابعی (function) مشاهده می کنید که سه نسخه ی مجزا دارد، بنابراین IntelliSense ۱ از ۳ را نمایش می دهد.



برای انتخاب یکی از ورژن ها، می توان کلید پیکان جهت پایین (down arrow key) را فشار داد یا مقدار آرگومان مورد نظر را تایپ کرده تا برنامه، نسخه ی مورد نظر را نمایش دهد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

آرگومان های اختیاری

آرگومانی با مقدار اختیاری

گفته شد در صورتی که متدی آرگومان گرفت، هنگام فراخوانی آن آرگومان، لازم است مقداری برای آرگومان نام برده فراهم کرد. البته، استثنایی برای قاعده ی مذکور وجود دارد. چنانچه متدی دارید که آرگومان آن معمولاً مقداری یکسان دریافت می کند، می توانید مقداری پیش فرض به آرگومان مزبور اختصاص دهید.

کد زیر را در نظر بگیرید.

```
public class Exercise
{
  double? CalculateNetPrice(double? discountRate)
  {
    double? origPrice = 125.55;
    return origPrice - (origPrice * discountRate / 100);
  }
  static int Main()
  {
    var exo = new Exercise();
    double? finalPrice = null;
    double? discount = 15; // That is 25% = 25
    finalPrice = exo.CalculateNetPrice(discount);
    System.Console.WriteLine("After applying the discount");
    System.Console.Write("Final Price = ");
    System.Console.WriteLine(finalPrice);
    System.Console.ReadKey();
    return 0;
  }
}
```

نتیجه

```
After applying the discount
Final Price = 106.7175
Press any key to continue...
```

برای نشان دادن این که آرگومانی مقدار پیش فرض دارد، مقدار پیش فرض را داخل پرانتز متد، پس از اسم آرگومان قرار دهید.

مثال

```
public class Exercise
{
  double? CalculateNetPrice(double? discountRate = 25)
  {
    double? origPrice = 125.55;
    return origPrice - (origPrice * discountRate / 100);
  }
}
```

می توان هنگام فراخوانی متد، یک مقدار به آرگومان ارسال کرد. چنانچه می خواهید مقدار پیش فرض را به کاربرید، دیگر نیازی به ارسال آرگومان نیست.

مثال

```
public class Exercise
{
  double? CalculateNetPrice(double? discountRate = 25)
  {
    double? origPrice = 125.55;
    return origPrice - (origPrice * discountRate / 100);
  }
  static int Main()
  {
    var exo = new Exercise();
    double? finalPrice = null;
    finalPrice = exo.CalculateNetPrice();
    System.Console.WriteLine("After applying the discount");
    System.Console.Write("Final Price = ");
    System.Console.WriteLine(finalPrice);
    System.Console.ReadKey();
    return 0;
  }
}
```

توجه داشته باشید که پرانتزهای متد خالی هستند، به این معنی که آرگومانی ارسال نشده است. نتیجه ی زیر به دست می آید.

```
After applying the discount
Final Price = 94.1625
Press any key to continue...
```

به همین ترتیب، می توان متدی ایجاد کرد که آرگومان های متعددی می گیرد. حال، کلیه ی آرگومان های مزبور (یا برخی از آن ها) می توانند مقدارهای پیش فرض داشته باشند.

در صورتی که متدی چند آرگومان (بیش از یک آرگومان) گرفت، می توان آرگومانی پیش فرض برای هر یک فراهم کرد، سپس انتخاب کرد کدام یک از آرگومان ها مقدار پیش فرض دریافت کنند. اگر می خواهید کلیه ی آرگومان ها مقدار پیش فرض داشته باشند، حین تعریف متد مورد نظر، به دنبال هر اسم علامت مساوی " = " و مقدار دلخواه را تایپ کنید. توجه خود را به مثال زیر جلب کنید.

```
public class Exercise
{
  double? CalculateNetPrice(double? tax = 5.75, double? discount = 25,double? origPrice = 245.55)
  {
    double? discountValue = origPrice * discount / 100;
    double? taxValue = tax / 100;
    double? netPrice = origPrice - discountValue + taxValue;
    System.Console.Write("Original Price: $");
    System.Console.WriteLine(origPrice);
    System.Console.Write("Discount Rate: ");
    System.Console.Write(discount);
    System.Console.WriteLine("%");
    System.Console.Write("Tax Amount: $");
    System.Console.WriteLine(tax);
    return netPrice;
  }
  static int Main()
  {
    var exo = new Exercise();
    double? finalPrice = null;
    finalPrice = exo.CalculateNetPrice();
    System.Console.WriteLine("After applying the discount");
    System.Console.Write("Final Price = ");
    System.Console.WriteLine(finalPrice);
    System.Console.ReadKey();
    return 0;
  }
}
```

نتیجه

```
Original Price: $245.55
Discount Rate: 25%
Tax Amount: $5.75
After applying the discount
Final Price = 184.22
Press any key to continue...
```

چنانچه متدی چند آرگومان (بیش از یک آرگومان) گرفت، و شما خواستید مقدارهای پیش فرضی برای آن پارامترها فراهم کنید، ترتیب قرار گیری آرگومان ها از اهمیت خاصی برخوردار خواهد بود.

اگر متد دو آرگومان داشت، می توان آن را با مقادیر پیش فرض تعریف کرد. چنانچه می خواهید مقداری پیش فرض برای تنها یکی از آرگومان ها ارائه دهید، آن مقدار پیش فرض فقط به آرگومان دومی تعلق می گیرد. هنگام فراخوانی چنین تابعی، چنانچه تنها یک آرگومان فراهم کنید، compiler مقدار آن را به اولین پارامتر در لیست تخصیص می دهد و هیچ مقداری به دومین پارامتر اختصاص نمی دهد.

```
public class Exercise
{
  double? CalculateNetPrice(double? taxRate, double? discountRate = 25)
  {
    double? origPrice = 185.95;
    double? discountValue = origPrice * discountRate / 100;
    double? taxValue = taxRate / 100;
    double? netPrice = origPrice - discountValue + taxValue;
    return netPrice;
  }
  static int Main()
  {
    double? finalPrice;
    double? taxRate = 5.50; // = 5.50%
    Exercise exo = new Exercise();
    finalPrice = exo.CalculateNetPrice(taxRate);
    System.Console.WriteLine("After applying a 25% discount and a 5.50% tax rate");
    System.Console.Write("Final Price = ");
    System.Console.WriteLine(finalPrice);
    System.Console.ReadKey();
    return 0;
  }
}
```

مثال

```
After applying a 25% discount and a 5.50% tax rate
Final Price = 139.5175
Press any key to continue...
```

چنانچه پس از تعریف تابع و تخصیص مقدار پیش فرض به اولین آرگومان، تنها یک آرگومان هنگام فراخوانی تابع فراهم کنید، با پیغام خطا مواجه می شوید. کد زیر را در نظر بگیرید.

```
public class Exercise
{
  double? CalculateNetPrice(double? discountRate = 25, double? taxRate)
```

```
  {
    double? origPrice = 185.95;
    double? discountValue = origPrice * discountRate / 100;
    double? taxValue = taxRate / 100;
    double? netPrice = origPrice - discountValue + taxValue;
    return netPrice;
  }
  static int Main()
  {
    double? finalPrice;
    double? taxRate = 5.50; // = 5.50%
    Exercise exo = new Exercise();
    finalPrice = exo.CalculateNetPrice(taxRate);
    System.Console.WriteLine("After applying a 25% discount and a 5.50% tax rate");
    System.Console.Write("Final Price = ");
    System.Console.WriteLine(finalPrice);
    System.Console.ReadKey();
    return 0;
  }
}
```

دو پیغام خطای زیر صادر می شوند.

```
Error 1 Optional parameters must appear after all required parameters
C:\Exercises\Bank\Exercise.cs 3   70   Bank
```

حال، چنانچه متد چندین آرگومان (بیش از دو آرگومان) دریافت کرد و می خواهید که تنها تعدادی از آن آرگومان ها مقدار پیش فرض داشته باشند، به خاطر داشته باشید که آرگومان های با مقدار پیش فرض باید انتهای لیست قرار بگیرند. صرف نظر از این که چه تعداد آرگومان مقدار پیش فرض دریافت می کنند (و چه تعداد مقدار پیش فرض دریافت نمی کنند)، همیشه لیست خود را با آرگومان هایی آغاز کنید که مقدار پیش فرض به کار نمی گیرند.

مثال

```
public class Exercise
{
  double? CalculateNetPrice(double? origPrice,double? taxRate = 5.75,double? discountRate = 25)
  {
    double? discountValue = origPrice * discountRate / 100;
    double? taxValue = taxRate / 100;
    double? netPrice = origPrice - discountValue + taxValue;
    return netPrice;
  }
}
```

در هنگام فراخوانی متد، به یاد داشته باشید که تنها برای آرگومان هایی که مقدار پیش فرض ندارند می توان مقدار ارائه داد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
public class Exercise
{
  double? CalculateNetPrice(double? origPrice,double? taxRate = 5.75,double? discountRate = 25)
  {
    double? discountValue = origPrice * discountRate / 100;
    double? taxValue = taxRate / 100;
    double? netPrice = origPrice - discountValue + taxValue;
    return netPrice;
  }
  static int Main()
  {
    double? originalPrice = 375.95;
    double? finalPrice;
    Exercise exo = new Exercise();
    finalPrice = exo.CalculateNetPrice(originalPrice);
    System.Console.WriteLine("After applying a 25% discount and a 5.75% tax rate");
    System.Console.Write("Final Price = ");
    System.Console.WriteLine(finalPrice);
    System.Console.ReadKey();
    return 0;
  }
}
```

مثال

```
After applying a 25% discount and a 5.75% tax rate
Final Price = 282.02
Press any key to continue...
```

فراخوانی آرگومان با (استفاده از) اسم

تصور کنید متدی دارید که چندین آرگومان اختیاری می گیرد. در C++، برای تخصیص مقدار به دومین آرگومان اختیاری، ابتدا باید مقداری برای اولین آرگومان اختیاری ارسال کنید. حتی اگر هم بخواهید مقدار پیش فرض آرگومان اولی را به کاربرید، باید آن را ارسال کنید. به مثال زیر توجه کنید.

```
public class Exercise
{
  double? CalculateNetPrice(double? origPrice,double? taxRate = 5.75,double? discountRate = 25)
  {
    double? discountValue = origPrice * discountRate / 100;
    double? taxValue = taxRate / 100;
    double? netPrice = origPrice - discountValue + taxValue;
    return netPrice;
  }
```

```
  static int Main()
  {
    double? finalPrice;
    Exercise exo = new Exercise();
    finalPrice = exo.CalculateNetPrice(375.95, 5.75, 40);
    System.Console.WriteLine("Original Price: 375.95");
    System.Console.WriteLine("Tax Rate:    5.75%");
    System.Console.WriteLine("Discount Rate: 40%");
    System.Console.Write("Final Price:  ");
    System.Console.WriteLine(finalPrice);
    System.Console.ReadKey();
    return 0;
  }
}
```

نتیجه ی زیر حاصل می گردد.

```
Original Price: 375.95
Tax Rate:    5.75%
Discount Rate: 40%
Final Price:  225.6275
Press any key to continue...
```

تصور کنید می خواهید یک آرگومان اختیاری را نادیده گرفته تا از مقدار پیش فرض آن استفاده کنید، ولی برای آرگومان های دوم، سوم.. می خواهید مقدار ارسال کنید. برای این منظور، در پرانتز متدی که فرا می خوانید، اسم آرگومان مورد نظر و به دنبال آن دو نقطه (:) و مقدار دلخواه را تایپ کنید (که در واقع با پروسه ی فراخوانی متد با استفاده از اسم یکی است).

مثال

```
public class Exercise
{
  double? CalculateNetPrice(double? origPrice,double? taxRate = 5.75,double? discountRate = 25)
  {
    double? discountValue = origPrice * discountRate / 100;
    double? taxValue = taxRate / 100;
    double? netPrice = origPrice - discountValue + taxValue;
    return netPrice;
  }
  static int Main()
  {
    double? finalPrice;
    Exercise exo = new Exercise();
    finalPrice = exo.CalculateNetPrice(discountRate: 40, origPrice: 375.95);
    System.Console.WriteLine("Original Price: 375.95");
```

```
    System.Console.WriteLine("Tax Rate:    5.75%");
    System.Console.WriteLine("Discount Rate: 40%");
    System.Console.Write("Final Price:  ");
    System.Console.WriteLine(finalPrice);
    System.Console.ReadKey();
    return 0;
  }
}
```

با استفاده از این روش، می توان آرگومان های اختیاری را که حاضر نیستید مقداری برای آن ها فراهم کنید نادیده گرفت.

مبانی سازنده ها

توصیف

کلاس زیر را در نظر بگیرید.

```
public class BankAccount
{
  private string customerName;
  private decimal originalDeposit;
}
```

پس از تعریف متغیر یک کلاس در برنامه، زمانی که برنامه بالا می آید، compiler به هر یک از اعضای کلاس به اندازه ی کافی حافظه اختصاص می دهد. فضای ای که به هر متغیر عضو تخصیص داده می شود با توجه به نوع آن متغیر مقداردهی اولیه می شود.

برای شی از نوع string (string object)، فضای مزبور خالی نگه داشته می شود. برای نوع integer، فضای حافظه با ۰ پر می شود. برای شئ ای از این نوع، بهتر است مقداری فراهم شود که متغیرهای عضو را با مقدارهای دلخواه شما مقداردهی اولیه می کند.

متدی که شی ای را مقداردهی (اولیه) می کند، ممکن است هر مقداری برگرداند، اما بهتر است مقدار بازگشتی از نوع void باشد زیرا هدف اصلی آن تنظیم دوباره ی مقادیر است. به این خاطر که متد، مقداری اولیه به تک تک متغیرهای عضو می دهد (متغیرهایی که طبیعتاً باید مقداردهی اولیه شوند)، لازم است آرگومانی معادل برای هر یک از متغیرهایی که مقداردهی اولیه می کند داشته باشد.

```
public class BankAccount
{
  private string customerName;
  private decimal originalDeposit;
  public void Initialize(string name, decimal deposit)
  {
  }
```

```
}
```

متدی که کلاسی را مقداردهی اولیه می کند، لزومی ندارد حتماً تمامی اعضای یک کلاس را مقداردهی (اولیه) کند. به منظور پیاده سازی متد (منظور متدی است که مقداردهی اولیه می کند)، کافی است آرگومان متد را به متغیر مربوط اختصاص دهید. به مثال های ذیل توجه کنید.

```
public class BankAccount
{
  private string customerName;
  private decimal originalDeposit;
  public void Initialize(string name, decimal deposit)
  {
    customerName = name;
    originalDeposit = deposit;
  }
}
```

سپس می توانید متد را پس از تعریف نمونه ای از کلاس، فراخوانی کنید تا مقادیر اولیه را به فیلدهای کلاس بدهد.

مثال

```
public class BankAccount
{
  private string customerName;
  private decimal originalDeposit;
  public void Initialize(string name, decimal deposit)
  {
    customerName = name;
    originalDeposit = deposit;
  }
  public void Show()
  {
    System.Console.WriteLine("Customer Account Information");
    System.Console.Write("Customer Name: ");
    System.Console.WriteLine(customerName);
    System.Console.Write("Original Deposit: ");
    System.Console.WriteLine(originalDeposit);
  }
}
public class Exercise
{
  static int Main()
  {
    BankAccount account = new BankAccount();
    account.Initialize("Paul Motto", 450.00M);
    account.Show();
    return 0;
```

```
    }
}
```

نتیجه

```
Customer Account Information
Customer Name: Paul Motto
Original Deposit: 450.00
Press any key to continue...
```

معرفی سازنده ها

۱. Microsoft Visual Studio را اجرا کنید.

۲. به منظور ایجاد برنامه ی جدید، فهرست گزینه ی اصلی برنامه را باز کرده، سپس گزینه های File -> New Project را کلیک کنید.

۳. گزینه ی Empty Project را از لیست میانی انتخاب کنید.

۴. اسم را به DepartmentStore3 تغییر داده، روی ok کلیک کنید.

۵. در پنجره ی Solution Explorer راست کلیک کرده، سپس طبق دستور زیر عمل کنید.

۶. DeparmentStore3 ->Add ->New Item.

۷. در لیست میانی پنجره ی محاوره ی Add New Item، Code File را کلیک کنید.

۸. حال، اسم را به StoreItem تغییر داده و سپس Add را کلیک کنید.

۹. در فایل دستورات زیر را وارد کنید.

```
public class StoreItem
{
   public long itemNumber;
   public string itemName;
   public string size;
   public decimal unitPrice;
}
```

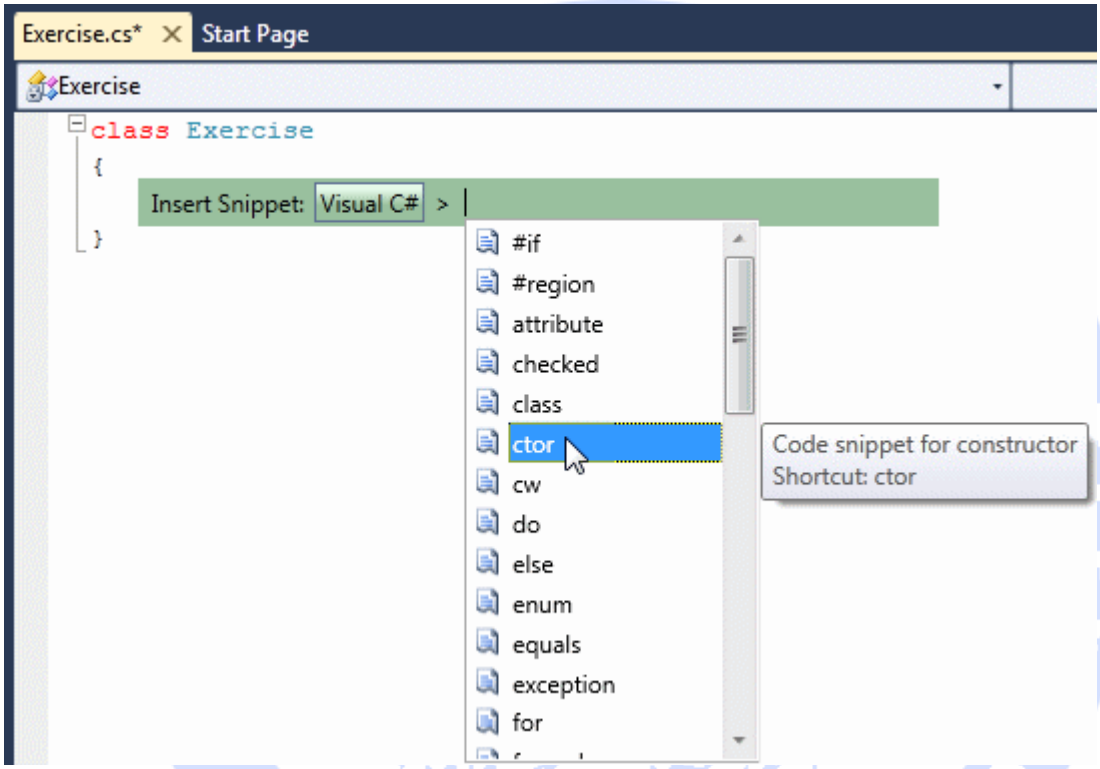
سازنده ی (constructor) پیش فرض

سازنده متدی خاص است که هنگام به وجود آمدن شی جدید (object) ساخته می شود. متد و کلاس هر دو یک اسم دارند. لازم به ذکر است متد، شی را درست در زمان به وجود آمدن آن مقداردهی اولیه می کند. هنگامی که کلاسی می سازید، چنانچه سازنده ای برای آن معرفی نکنید، compiler به صورت خودکار یک سازنده برای شما ایجاد می کند؛ این کار باعث می شود که کلیه ی شی

های کلاس از وجود شی جدید آگاه شوند. به سازنده ای که توسط compiler ایجاد می شود، سازنده ی پیش فرض نیز می گویند.

البته شما می توانید سازنده ی خود را ایجاد کنید. برای این منظور متدی معرفی کنید که با کلاس هم نام است.

داخل کلاس موردنظر راست کلیک کرده Insert Snippet… را انتخاب کنید، سپس C# Visual را دوبار کلیک کنید. حال، در لیستی که ظاهر می شود روی ctor دوبار کلیک کنید.



Code Editor برای ایجاد متد جدید از اسم کلاس استفاده می کند

به خاطر داشته باشید که متد فوق نباید هیچ مقداری بازگرداند.

مثال

```
public class BankAccount
{
  public BankAccount()
  {
  }
}
```

زمانی که نمونه ای از کلاس تعریف می کنید، (چه از آن شی استفاده بکنید چه نکنید)، سازنده ای برای شی ایجاد می شود. پس از این که نمونه ای از کلاس تعریف شد، سازنده ی پیش فرض فراخوانده می شود (چه شی نام برده را به کار ببرید و چه آن را نادیده بگیرید). مثال زیر به زیبایی عملیات ذکر شده را تشریح می کند.

```
public class BankAccount
{
  public BankAccount()
  {
    System.Console.WriteLine("New Bank Account");
  }
}
public class Exercise
{
  static int Main()
  {
    BankAccount account = new BankAccount();
    return 0;
  }
}
```

نتیجه

```
New Bank Account
Press any key to continue...
```

همان طور که در مثال فوق مشاهده کردید، اگرچه شی مذکور مورد استفاده قرار نگرفت، تعریف آن به تنهایی وجود شی جدید را نشان داد.

به کاربردن سازنده ی پیش فرض

۱. در پنجره ی Solution Explorer، راست کلیک کرده، سپس DepartmentStore3 ->Add ->New Item...

۲. در لیست میانی پنجره ی محاوره ی Add New Item، روی Code File کلیک کنید. حال، اسم را به DepartmentStore تغییر داده و Add را کلیک کنید

۳. فایل را به صورت زیر اصلاح کنید.

```
public class DepartmentStore
{
  static int Main()
  {
```

```
    StoreItem si = new StoreItem();
    System.Console.WriteLine("Department Store");
    System.Console.Write("Item #:  ");
    System.Console.WriteLine(si.itemNumber);
    System.Console.Write("Item Name: ");
    System.Console.WriteLine(si.itemName);
    System.Console.Write("Item Size: ");
    System.Console.WriteLine(si.size);
    System.Console.Write("Unit Price: ");
    System.Console.WriteLine(si.unitPrice);
    System.Console.ReadKey();
    System.Console.ReadKey();
    return 0;
  }
}
```

برنامه را اجرا کنید. این نتیجه را به دست می دهد.

```
Department Store
Item #:  0
Item Name:
Item Size:
Unit Price: 0
```

۵. پنجره ی DOS را بسته و به محیط برنامه نویسی باز گردید.

سازنده ای که مقداردهی اولیه می کند

می توان از سازنده (constructor) برای مقداردهی اولیه ی فیلدهای یک کلاس نیز استفاده کرد. به همین دلیل، می تواند جایگزین مناسبی برای متدی باشد که مقداردهی اولیه می کند. برای استفاده از سازنده به منظور مقداردهی اولیه ی فیلدهای یک کلاس، متغیرهایی را که می خواهید مقداردهی اولیه کنید را به عنوان آرگومان تعریف کنید. نیازی نیست تمامی متغیرهای عضو را تعریف کنید، بلکه تنها آن دسته متغیرهایی که ملزوم به مقداردهی اولیه هستید کفایت می کند. در حقیقت، تنها باید آن عضوهایی را مقداردهی (اولیه) کنید که مطمئن هستید، دیگر شی ها به هنگام استفاده از این شئ به آن ها نیاز دارند. به عبارت روشن تر، object شما فیلدهایی دارد که اشیا خارجی (external objects) نیاز ندارند تغییر دهند (یا به آن ها دسترسی پیدا کنند) یا متغیرهای عضو بعده ها هنگامی که از شئ مورد نیاز فراخوانده می شوند، مقداردهی اولیه می شوند.

به منظور ایجاد سازنده ی پیش فرض، می توانید فقط فیلدهای دلخواه کلاس را مقداردهی اولیه کنید. برای متغیری از نوع عددی، کافی است ثابت دلخواه را به هر متغیر عضو اختصاص دهید. در صورتی که متغیر یک کاراکتر بود، فقط یک تک کوتیشن(') به آن اضافه می کنیم. حال، چنانچه متغیر یک رشته بود، باید مقداری با (") دابل کوتیشن به متغیر اختصاص داد.

```
public class BankAccount
{
  private string customerName;
  private decimal originalDeposit;
  public BankAccount()
  {
    customerName = "John Doe";
    originalDeposit = 0M;
  }
  public void Show()
  {
    System.Console.WriteLine("Customer Account Information");
    System.Console.Write("Customer Name: ");
    System.Console.WriteLine(customerName);
    System.Console.Write("Original Deposit: ");
    System.Console.WriteLine(originalDeposit);
  }
}
public class Exercise
{
  static int Main()
  {
    BankAccount account = new BankAccount();
    account.Show();
    System.Console.ReadKey();
    return 0;
  }
}
```

نتیجه

```
Customer Account Information
Customer Name: John Doe
Original Deposit: 0
Press any key to continue...
```

دستورالعمل ایجاد سازنده ای که مقداردهی (اولیه) می کند

۱. فهرست گزینه ی اصلی را باز کرده، روی گزینه ی Window ->StoreItem.cs کلیک کنید.

۲. فایل را به صورت زیر اصلاح کنید.

```
public class StoreItem
{
  public long itemNumber;
  public string itemName;
  public string size;
  public decimal unitPrice;
  public StoreItem()
  {
    itemNumber = 0;
    itemName = "Unknown";
    size = "0";
    unitPrice = 0.00M;
  }
}
```

۳. برنامه را اجرا کنید. نتیجه ی زیر حاصل می گردد.

```
Department Store
Item #: 0
Item Name: Unknown
Item Size: 0
Unit Price: 0.00
```

۴. پنجره ی DOS را بسته و به محیط برنامه نویسی بازگردید.

سربارگذاری سازنده (constructor overloading)

مناسب ترین مکان برای تعریف اولیه ی متغیر ها درون سازنده می باشد (مناسب ترین جا برای تخصیص مقادیر پیش فرض به اعضای کلاس، سازنده ی پیش فرض می باشد). علاوه بر سازنده ی پیش فرض، می توانید هر تعداد سازنده که ملزوم می دانید اضافه کنید. این جنبه به شما اجازه می دهد برای اهداف مختلف، سازنده های متفاوت تولید کنید. بنابراین، یک سازنده نیز (مثل متد) امکان overload شدن دارد. در مورد کلیه ی قواعد سربارگذاری متد قبلاً مفصل بحث کردیم. ابتدایی ترین سازنده ای که ایجاد می کنید، قابلیت استفاده از یک آرگومان را دارد. هنگام ایجاد سازنده ای که تنها یک آرگومان می گیرد، باید آن عضوی را مقداردهی اولیه کنید که با آرگومان مذکور مطابقت دارد، و دیگر اعضا را با مقادیر پیش فرض مقداردهی کنید. به مثال زیر توجه کنید.

```
public class BankAccount
{
  private string accountNumber;
  private string customerName;
```

```
    private decimal originalDeposit;
    public BankAccount(string number)
    {
      accountNumber = number;
      customerName = "John Doe";
      originalDeposit = 0M;
    }
    public void Show()
    {
      System.Console.WriteLine("Customer Account Information");
      System.Console.Write("Account N#:    ");
      System.Console.WriteLine(accountNumber);
      System.Console.Write("Customer Name:  ");
      System.Console.WriteLine(customerName);
      System.Console.Write("Original Deposit: ");
      System.Console.WriteLine(originalDeposit);
      System.Console.ReadKey();
    }
}
```

در صورتی که کلاسی با تنها یک سازنده ایجاد کردید (مانند مثال فوق)، باید هنگام معرفی نمونه ای از کلاس، سازنده ی نام برده را به کار ببرید: توجه داشته باشید که نمی توان از سازنده ی پیش فرض ای که آرگومان نمی گیرد استفاده کرد. هنگام تعریف متغیر، آن را با سازنده مقداردهی اولیه کنید، سپس مقادیر را داخل پرانتز سازنده قرار دهید.

```
public class BankAccount
{
  private string accountNumber;
  private string customerName;
  private decimal originalDeposit;
  public BankAccount(string number)
  {
    accountNumber = number;
    customerName = "John Doe";
    originalDeposit = 0M;
  }
  public void Show()
  {
    System.Console.WriteLine("Customer Account Information");
    System.Console.Write("Account #:    ");
    System.Console.WriteLine(accountNumber);
    System.Console.Write("Customer Name:  ");
    System.Console.WriteLine(customerName);
    System.Console.Write("Original Deposit: ");
    System.Console.WriteLine(originalDeposit);
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
  }
}
public class Exercise
{
  static int Main()
  {
    BankAccount account = new BankAccount("27-940025-17");
    account.Show();
    System.Console.ReadKey();
    return 0;
  }
}
```

نتیجه ی زیر حاصل می گردد.

```
Customer Account Information
Account #:  27-940025-17
Customer Name: John Doe
Original Deposit: 0
Press any key to continue...
```

به همین ترتیب، می توان برای مقداردهی های اولیه ی متفاوت، سازنده های متفاوت ایجاد کرد، البته این امکان وجود ندارد که برای هر متغیر سازنده ای متفاوت ایجاد کرد. چنانچه سازنده ای متفاوت با آرگومان های مختلف (برای مقداردهی اولیه) ایجاد کنید، هنگام تعریف متغیرهای کلاس، دقت کنید که هر نمونه را با تعداد صحیح آرگومان مقداردهی اولیه کنید، در غیر این صورت، compiler ایراد می گیرد.

چنانچه کلاسی با تنها یک سازنده ایجاد کنید، و آن سازنده نیز حداقل یک آرگومان داشته باشد، سازنده ی پیش فرض دیگر قابل استفاده و در دسترس نخواهد بود. برای دسترسی به سازنده ی پیش فرض یک شی، دو گزینه پیش رو دارید.

- در صورتی که هیچ سازنده ای در کلاس ایجاد نکنید، سازنده ی پیش فرض همیشه هنگام فراخوانی کلاس آماده هست.
- اگر حداقل یک سازنده برای کلاس ایجاد کنید و حداقل یک آرگومان برای سازنده ی مزبور فراهم کنید، باید خودتان (صراحتا) یک سازنده ی پیش فرض برای کلاس ایجاد کنید.

سربارگذاری سازنده

۱. به فهرست گزینه ی اصلی مراجعه کرده، روی گزینه های Window ->StoreItem.cs کلیک کنید.

۲. فایل را به صورت زیر اصلاح کنید.

```
public class StoreItem
  {
    public long itemNumber;
    public string itemName;
    public string size;
    public decimal unitPrice;
    public StoreItem()
    {
      itemNumber = 0;
      itemName = "Unknown";
      size = "0";
      unitPrice = 0.00M;
    }
    public StoreItem(long number)
    {
      itemNumber = number;
      itemName = "Unknown";
      size = "0";
      unitPrice = 0.00M;
    }
    public StoreItem(long number, string name, string itemSize, decimal price)
    {
      itemNumber = number;
      itemName = name;
      size = itemSize;
      unitPrice = price;
    }
  }
```

۳. به فایل DepartmentStore.cs دسترسی پیدا کرده، سپس آن را به صورت زیر تغییر دهید.

```
public class DepartmentStore
  {
    static int Main()
    {
      // Using the default contructor to create an unknown object
      StoreItem unknown = new StoreItem();
      System.Console.WriteLine("Department Store");
      System.Console.Write("Item #:  ");
      System.Console.WriteLine(unknown.itemNumber);
      System.Console.Write("Item Name: ");
      System.Console.WriteLine(unknown.itemName);
      System.Console.Write("Item Size: ");
      System.Console.WriteLine(unknown.size);
      System.Console.Write("Unit Price: ");
      System.Console.WriteLine(unknown.unitPrice);
      System.Console.WriteLine("=----------------------------------------=");
```

```
    // Using the contructor that takes one argument create an object
    // whose only known information is the item number
    StoreItem knownNumber = new StoreItem(227174);
    System.Console.WriteLine("Department Store");
    System.Console.Write("Item #:  ");
    System.Console.WriteLine(knownNumber.itemNumber);
    System.Console.Write("Item Name: ");
    System.Console.WriteLine(knownNumber.itemName);
    System.Console.Write("Item Size: ");
    System.Console.WriteLine(knownNumber.size);
    System.Console.Write("Unit Price: ");
    System.Console.WriteLine(knownNumber.unitPrice);
    System.Console.WriteLine("=----------------------------------------=");
    // Using the contructor that has all the information
    // necessary to create an object
    StoreItem complete = new StoreItem(180318, "V-Neck Cardigan with Ruffle Trim", "M", 74);
    System.Console.WriteLine("Department Store");
    System.Console.Write("Item #:  ");
    System.Console.WriteLine(complete.itemNumber);
    System.Console.Write("Item Name: ");
    System.Console.WriteLine(complete.itemName);
    System.Console.Write("Item Size: ");
    System.Console.WriteLine(complete.size);
    System.Console.Write("Unit Price: ");
    System.Console.WriteLine(complete.unitPrice);
    System.Console.WriteLine("=----------------------------------------=");
    System.Console.ReadKey();
    return 0;
  }
}
```

۴. برنامه را اجرا کنید. نتیجه ی زیر به دست می آید.

```
Department Store
Item #:  0
Item Name: Unknown
Item Size: 0
Unit Price: 0.00
=----------------------------------------=
Department Store
Item #:  227174
Item Name: Unknown
Item Size: 0
Unit Price: 0.00
=----------------------------------------=
Department Store
Item #:  180318
Item Name: V-Neck Cardigan with Ruffle Trim
```

```
Item Size: M
Unit Price: 74
=-----------------------------------------=
```

۵. پنجره ی DOS را بسته و به محیط برنامه نویسی بازگردید.

سازنده ای با مقادیر پیش فرض

به این خاطر که یک سازنده پیش از هر چیز یک متد است و می تواند آرگومان بگیرد،متعاقباً آرگومان های آن می توانند مقادیر پیش فرض به کاربرند.

به منظور تخصیص مقدار پیش فرض به آرگومان یک سازنده، مقدار مناسب را هنگام ایجاد سازنده به آرگومان اختصاص دهید.

مثال

```
public class Rectangle
{
  private double len;
  private double hgt;
  public Rectangle(double side = 10.00D)
  {
  }
}
```

می توان از سازنده برای مقداردهی اولیه ی فیلدهای یک کلاس استفاده کرد.

اگر سازنده ای ایجاد کنید که یک آرگومان می گیرد، هنگام ایجاد نمونه ای از کلاس، آن تک سازنده هم به عنوان یک سازنده ی پیش فرض عمل می کند، هم به عنوان سازنده ای که یک آرگومان می گیرد. به این معنا که شما می توانید متغیری تعریف کرده و از سازنده ای استفاده کنید که پرانتزهای آن خالی است. به مثال زیر توجه کنید.

```
public class Rectangle
  {
    private double len;
    private double hgt;
    // Length = Height: Square
    public Rectangle(double side = 10.00D)
    {
      len = side;
      hgt = side;
    }
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    public void Describe()
    {
      System.Console.WriteLine("Square Characteristics");
      System.Console.Write("Side: ");
      System.Console.WriteLine(len);
    }
  }
  public class Exercise
  {
    static int Main()
    {
      Rectangle rect = new Rectangle();
      rect.Describe();
      System.Console.ReadKey();
      return 0;
    }
  }
```

نتیجه ی زیر به دست می آید.

```
Square Characteristics
Side: 10
Press any key to continue...
```

به همین ترتیب می توانید سازنده های مختلفی ایجاد کنید که آرگومان های متفاوتی می گیرند، و برخی از آرگومان های ذکر شده می توانند مقادیر پیش فرض داشته باشند. سازنده های گوناگون می توانند آرگومان های از نوع متفاوت (مختلفی) داشته باشند، برخی آرگومان ها می توانند مقادیر پیش فرض داشته باشند، در حالی که برخی دیگر (از این آرگومان ها) مقادیر پیش فرض نداشته باشند. هنگام ایجاد شی، لازم است به نوع سازنده ای که به کار می برید دقت کنید. مثال زیر را در نظر بگیرید.

```
public class Rectangle
  {
    private double len;
    private double hgt;
    /* If the constructor is called with only one argument,
     * we are dealing with a square.
     * If the constructor is called with two arguments,
     * then we have a rectangle */
    public Rectangle(double length, double height = 10.00D)
    {
      len = length;
      hgt = height;
    }
    /* If the constructor is called with only one argument,
     * we are dealing with a square.*/
```

```
    public void Describe()
    {
      System.Console.WriteLine("Square Characteristics");
      System.Console.Write("Side: ");
      System.Console.WriteLine(len);
    }
    /* If the constructor is called with two arguments,
     * then we have a rectangle */
    public void Describe(int rect)
    {
      System.Console.WriteLine("Rectangle Characteristics");
      System.Console.Write("Length: ");
      System.Console.WriteLine(len);
      System.Console.Write("Height: ");
      System.Console.WriteLine(hgt);
    }
  }
  public class Exercise
  {
    static int Main()
    {
      Rectangle rect = null;
      rect = new Rectangle(24.72);
      rect.Describe();
      rect = new Rectangle(24.72, 20.64);
      rect.Describe(1000);
      System.Console.ReadKey();
      return 0;
    }
  }
```

نتیجه ی زیر حاصل می شود.

```
Square Characteristics
Side: 24.72
Rectangle Characteristics
Length: 24.72
Height: 20.64
Press any key to continue...
```

استفاده از سازنده های یک کلاس

مخرب های کلاس (class destructor)

درست برخلاف سازنده ها، یک مخرب زمانی فراخوانده می شود که برنامه دیگر کارش با شی تمام شده و نیازی به آن ندارد. مخرب فرایند پاک سازی را پشته صحنه انجام می دهد. درست مثل سازنده ی پیش فرض، در صورتی که مخربی برای کلاس تعریف نکنید، compiler خود

یک مخرب پیش فرض ایجاد می کند. برخلاف سازنده، مخرب را نمی توان سربارگذاری (overload) کرد. به عبارت دیگر، شما نمی توانید بیش از یک مخرب داشته باشید. اما درست مثل سازنده، مخرب با کلاس هم اسم است. این بار اسم مخرب با علامت "~" آغاز می شود.

برای ایجاد مخرب، علامت ~ و به دنبال آن اسم کلاس را تایپ کنید.

مثال

```
public class BankAccount
{
  private string accountNumber;
  private string customerName;
  private decimal originalDeposit;
  public BankAccount(string number)
  {
    accountNumber = number;
    customerName = "John Doe";
    originalDeposit = 0M;
  }
  public void Show()
  {
    System.Console.WriteLine("Customer Account Information");
    System.Console.Write("Account #:    ");
    System.Console.WriteLine(accountNumber);
    System.Console.Write("Customer Name:  ");
    System.Console.WriteLine(customerName);
    System.Console.Write("Original Deposit: ");
    System.Console.WriteLine(originalDeposit);
  }
  ~BankAccount()
  {
  }
}
```

متغیرهای خواندنی (read-only variables)

هنگام ایجاد متغیر عضو کلاس، یکی از تصمیماتی که اتخاذ می کنید این است که فیلد کلاس چگونه مقدارش را دریافت کند. گاهی اوقات به آن اعضایی (clients of a class) که از کلاس استفاده می کنند، اجازه ی تغییر مقادیر فیلدها داده می شود. گاهی اوقات هم شما می خواهید فیلد، فقط مقدار مزبور را نگه دارد یا ارائه بدهد، ولی در عین حال نتواند آن را تغییر دهد. این کار باز هم به اعضا اجازه می دهد که به فیلد دسترسی پیدا کنند ولی تنها در حد خواندن آن.

به منظور ایجاد فیلدی که مقدار آن را فقط بتوان خواند، پیش از نوع داده ی آن باید کلیدواژه ی readonly را تایپ کرد. به مثال زیر توجه کنید.

```
public readonly double PI;
```

پس از تعریف متغیر، لازم است آن را مقداردهی اولیه کنید. دو گزینه اصلی پیش رو دارید، می توانید فیلد را هنگام تعریف مقداردهی اولیه کنید.

مثال

```
public class Circle
{
  public double radius;
  public Circle(double rad)
  {
    radius = rad;
  }
  public readonly double PI = 3.14159;
}
public class Exercise
{
  static int Main()
  {
    var circ = new Circle(24.72);
    System.Console.WriteLine("Circle Characteristics");
    System.Console.Write("Radius: ");
    System.Console.WriteLine(circ.radius);
    System.Console.Write("PI:   ");
    System.Console.WriteLine(circ.PI);
    System.Console.ReadKey();
    return 0;
  }
}
```

نتیجه ی زیر را به دست می دهد.

```
Circle Characteristics
Radius: 24.72
PI:   3.14159
Press any key to continue...
```

گزینه ی دوم این است که فیلد را داخل سازنده ی یک کلاس مقداردهی اولیه کنید. این کار به صورت زیر انجام می پذیرد.

```
public class Circle
{
  public double radius;
  public Circle(double rad)
  {
    radius = rad;
    PI = 3.14159;
  }
  public readonly double PI;
}
public class Exercise
{
  static int Main()
  {
    var circ = new Circle(24.72);
    System.Console.WriteLine("Circle Characteristics");
    System.Console.Write("Radius: ");
    System.Console.WriteLine(circ.radius);
    System.Console.Write("PI:   ");
    System.Console.WriteLine(circ.PI);
    System.Console.ReadKey();
    return 0;
  }
}
```

چنانچه مقدار یک فیلد خواندنی (read-only field) از یک عبارت گرفته شده باشد، فیلد نام برده باید در سازنده و با عبارت دلخواه مقداردهی اولیه شود. بر این اساس، کد زیر ترجمه (compile) نمی‌شود.

```
public class Circle
{
  public double radius;
  public Circle(double rad)
  {
    radius = rad;
    PI = 3.14159;
  }
  public readonly double PI;
  public readonly double Diameter = radius * 2;
}
public class Exercise
{
  static int Main()
  {
    var circ = new Circle(24.72);
    System.Console.WriteLine("Circle Characteristics");
    System.Console.Write("Radius:  ");
    System.Console.WriteLine(circ.radius);
    System.Console.Write("PI:    ");
    System.Console.WriteLine(circ.PI);
```

```
        System.Console.Write("Diameter: ");
        System.Console.WriteLine(circ.Diameter);
        System.Console.ReadKey();
        return 0;
    }
}
```

نتیجه ی زیر به دست می آید.

```
Error   1       A field initializer cannot reference the non-static
field, method, or property 'Circle.radius'
C:\Exercise1\Exercise.cs                12              39              Bank
```

یکی از راه حل های رفع خطای فوق، این است که فیلد مزبور را به عنوان متغیری فقط خواندنی (read-only) در کلاس تعریف کرده، سپس آن را درسازنده و با عبارت موردنظر مقداردهی اولیه کنید . نمونه های آن را زیر مشاهده می کنید.

```
public class Circle
{
    public double radius;
    public Circle(double rad)
    {
        radius = rad;
        Diameter = radius * 2;
        Circumference = Diameter * PI;
        Area = radius * radius * PI;
    }
    public readonly double PI = 3.14159;
    public readonly double Diameter;
    public readonly double Circumference;
    public readonly double Area;
}
public class Exercise
{
    static int Main()
    {
        Circle circ = new Circle(24.72);
        System.Console.WriteLine("Circle Characteristics");
        System.Console.Write("Radius:      ");
        System.Console.WriteLine(circ.radius);
        System.Console.Write("Diameter:    ");
        System.Console.WriteLine(circ.Diameter);
        System.Console.Write("Circumference: ");
        System.Console.WriteLine(circ.Circumference);
        System.Console.Write("Area:        ");
        System.Console.WriteLine(circ.Area);
        System.Console.ReadKey();
```

```
    return 0;
  }
}
```

نتیجه ی زیر را به دست می دهد.

```
Circle Characteristics
Radius:  24.72
Diameter:  49.44
Circumference: 155.3202096
Area:  1919.757790656
Press any key to continue...
```

می دانیم که یک متغیر ثابت (constant variable) را باید به محض ایجاد کردن، مقداردهی کرد. لازم به ذکر است که متغیر (فقط) خواندنی از این قاعده پیروی نمی کند. به خاطر داشته باشید که نیازی نیست متغیر خواندنی را هنگام ایجاد آن مقداردهی اولیه کرد، زیرا این کار داخل سازنده ی کلاس صورت می گیرد. هم چنین به دلیل این که سازنده قابلیت overload شدن را دارد، می تواند مقادیر متفاوتی داشته باشد، اما مقدار متغیر ثابت هیچ گاه تغییر نمی کند : یک بار در کلاس (یا متد) مقداردهی می شود و آن مقدار را برای همیشه سرتاسر کلاس (یا متد) نگه می دارد (در تمام بخش های کلاس از همان مقدار استفاده می کند).

معرفی فضاهای نام (Namespace)

فضای نام بخشی از یک کد است، که با نام به خصوصی شناسایی می شود. آن نام می تواند هر اسمی از جمله اسم یک شخص، شرکت یا شهر باشد.

مقدمه

۱. ابتدا برنامه ی Microsoft Visual Studio را اجرا کنید.

۲. به منظور ایجاد پروژه ی جدید، به فهرست گزینه ی اصلی مراجعه کرده، سپس روی گزینه های File -> New Project کلیک کنید.

۳. در فهرست سمت چپ، Windows را انتخاب کنید.

۴. حال، روی Empty Project در لیست سمت راست کلیک کنید.

۵. اسم را به DepartmentStore4 تغییر دهید.

۶. اکنون، ok را کلیک کنید.

۷. برای ایجاد فایل جدید، فهرست گزینه ی اصلی را باز کرده، روی گزینه های Project -> Add New Item... کلیک کنید.

۸. گزینه ی Code را از لیست سمت چپ انتخاب کنید.

۹. در لیست سمت راست، روی Code File کلیک کنید.

۱۰. اسم را به DepartmentStore تغییر دهید، Add را کلیک کنید.

۱۱. در فایل خالی دستورات زیر را تایپ کنید.

```
public class DepartmentStore
{
  static int Main()
  {
    return 0;
  }
}
```

ایجاد فضای نام به صورت دستی

برای ایجاد فضای نام

کد را با کلیدواژه ی namespace آغاز کرده و به دنبال آن اسم بخش موردنظر را تایپ کنید.

روی بخشی که می خواهید فضای نام در آنجا ایجاد شود راست کلیک کرده، حال روی Insert Snippet... دوبار کلیک کنید.

درست مثل یک کلاس، بخشی که متعلق به فضای نام است با علامت "{" آغاز شده و به "}" ختم می شود.

مثال

```
namespace Business
{
}
```

داخل {} می توان هر چه مربوط به فضای نام است قرار داد. برای مثال، می توان داخل فضای نامی یک کلاس ایجاد کنید. به مثال زیر توجه کنید.

```
namespace Business
{
  class House
  {
  }
}
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

نحوه ی ایجاد فضای نام

برای این منظور، فایل را به صورت زیر تغییر دهید.

```
public class DepartmentStore
{
  static int Main()
  {
    return 0;
  }
}
namespace Store
{
  public class StoreItem
  {
    public int itemNumber;
    public string itemName;
    public decimal unitPrice;
  }
}
```

فضای نامی که به صورت اتوماتیک ایجاد شده

برای ایجاد کلاسی جدید به فهرست گزینه ی اصلی مراجعه کرده، روی Project -> Add Class... کلیک کنید.

در پنجره ی Solution Explorer، روی اسم پروژه ی موردنظر راست کلیک کرده سپس : project -> Add -> click Class... .

در پنجره ی Class View، روی اسم پروژه راست کلیک کرده سپس : Project -> Add -> Class...

در صورت استفاده از هریک روش های بالا، برنامه ی Microsoft Visual C# 2010 Express یا Microsoft Visual Studio فضای نامی با اسم پروژه ایجاد کرده، سپس کلاس جدید را به آن اضافه می کند.

دسترسی به اعضای فضای نام

پس از ایجاد اعضای یک فضای نامی، می توانید با استفاده از عملگر نقطه (.) به آیتم موردنظر در فضای نام دسترسی پیدا کنید. برای این منظور، اسم فضای نام و به دنبال آن عملگر نقطه (.) و عضو موردنظر فضای نام را تایپ کنید.

مثال

```
namespace Business
{
  public class House
  {
```

```
    public string propertyNumber;
    public decimal price;
  }
}
public class Exercise
{
  static void Main()
  {
    Business.House property = new Business.House();
    property.propertyNumber = "D294FF";
    property.Price = 425880;
  }
}
```

دسترسی به اعضای فضای نام

۱. برای دسترسی به محتوای فضای نام، فایل را به صورت زیر اصلاح کنید.

```
public class DepartmentStore
{
  static int Main()
  {
    Store.StoreItem si = new Store.StoreItem();
    si.itemNumber = 613508;
    si.itemName = "Merino Crew Neck Cardigan";
    si.unitPrice = 80.00M;
    System.Console.WriteLine("Store Inventory");
    System.Console.Write("Item #:  ");
    System.Console.WriteLine(si.itemNumber);
    System.Console.Write("Item Name: ");
    System.Console.WriteLine(si.itemName);
    System.Console.Write("Unit Price: ");
    System.Console.WriteLine(si.unitPrice);
    System.Console.ReadKey();
    return 0;
  }
}
namespace Store
{
  public class StoreItem
  {
    public int itemNumber;
    public string itemName;
    public decimal unitPrice;
  }
}
```

۲. برنامه را اجرا کنید.

```
Store Inventory
 Item #:  613508
 Item Name: Merino Crew Neck Cardigan
 Unit Price: 80.00
```

۳. کلید Enter را زده تا به محیط برنامه نویسی باز گردید.

استفاده از چندین فضای نام

مقدمه

در مثال بالا، تنها یک فضای نام ایجاد کردیم. به همین شکل، می توانید فضاهای نامی متعددی در فایل ایجاد کنید، البته به شرط اینکه بدنه ی تمامی آن ها به درستی مرتب (delimit) شده باشد.

مثال

```
namespace RealEstate
{
  public class House
  {
    public string propertyNumber;
    public decimal price;
  }
}
namespace Dealership
{
  public class Car
  {
  }
}
```

همچنین می توان فضاهای نامی متعدد در چندین فایل ساخت. پس از ایجاد فایل ها، برای دسترسی به محتوای فضای نام، لازم است اسم کلاس را تعریف کنید.

ایجاد چندین فضای نام

۱. برای ایجاد فایل جدید، فهرست اصلی را باز کرده، روی Project -> Add New Item... کلیک کنید.

۲. در صورت نیاز، در لیست سمت چپ روی Code کلیک کنید. حال، در لیست سمت راست روی Code File کلیک کنید.

۳. اسم را به Records تغییر داده، سپس Add را کلیک کنید.

۴. فایل را به این شکل زیر تغییر دهید.

```
namespace Store
{
  public class StoreItem
  {
    public int itemNumber;
    public string itemName;
    public decimal unitPrice;
  }
}
```

۵. برای ایجاد فایل جدید، فهرست اصلی را باز کرده سپس : Project -> Add New Item...

۶. در صورت لزوم، گزینه ی Code را از لیست سمت چپ انتخاب کنید. اکنون، در لیست سمت راست روی Code File کلیک کنید.

۷. اسم را به Suppliers تغییر داده، سپس روی Add کلیک کنید.

۸. فایل را به صورت زیر اصلاح کنید.

```
namespace Supply
{
  public class Manufacturer
  {
    public string companyName;
    public string contactName;
    public string contactPhone;
  }
}
```

۹. برای دستیابی به فایل دیگر، روی برچسب DepartmentStore.cs کلیک کنید.

۱۰. برای به کار بردن فضاهای نامی، فایل را به صورت زیر اصلاح کنید.

```
public class DepartmentStore
{
  static int Main()
  {
    Supply.Manufacturer dealer = new Supply.Manufacturer();
    dealer.companyName = "Peel Corp";
    dealer.contactName = "Sylvain Yobo";
    dealer.contactPhone = "(602) 791-8074";
```

```
    Store.StoreItem si = new Store.StoreItem();
    si.itemNumber = 613508;
    si.itemName = "Merino Crew Neck Cardigan";
    si.unitPrice = 80.00M;
    System.Console.WriteLine("Store Inventory");
    System.Console.Write("Item #:  ");
    System.Console.WriteLine(si.itemNumber);
    System.Console.Write("Item Name: ");
    System.Console.WriteLine(si.itemName);
    System.Console.Write("Unit Price: ");
    System.Console.WriteLine(si.unitPrice);
    System.Console.ReadKey();
    return 0;
  }
}
```

استفاده از (یک) فضای نام

دیدیم که به منظور فراخوانی شی یا متدی که بخشی از فضای نام می باشد، باید شی مورد نظر را با عملگر نقطه (.) تعریف کنید. برای استفاده از namespace ای که قبلاً در فایل دیگری ایجاد شده و مورد استفاده قرار گرفته، لازم است از کلیدواژه ی using استفاده کنید (با این کار نشان می دهید که فضای نامی که اکنون دارید به کار می برید، قبلاً در جای دیگری تعریف شده). برای انجام این کار، ابتدا کلیدواژه ی using و به دنبال آن اسم فضای نامی را تایپ کنید.

لازم به ذکر است که با استفاده از کلیدواژه ی using می توان هر تعداد فضای نام خارجی که لازم است دخیل کرد.

در صورت بروز تداخل اسم، حتی اگر هم از کلیدواژه ی using استفاده می کنید، باز هم باید اسم کلاس را کامل تعریف کنید.

به کار بردن فضاهای نام

۱. برای استفاده از کلیدواژه ی using، فایل را به صورت زیر تغییر دهید.

```
public class DepartmentStore
{
  static int Main()
  {
    Manufacturer dealer = new Manufacturer();
    dealer.companyName = "Peel Corp";
    dealer.contactName = "Sylvain Yobo";
    dealer.contactPhone = "(602) 791-8074";
    StoreItem si = new StoreItem();
    si.itemNumber = 613508;
    si.itemName = "Merino Crew Neck Cardigan";
```

```
    si.unitPrice = 80.00M;
    System.Console.WriteLine("Manufacturer Information");
    System.Console.Write("Company Name: ");
    System.Console.WriteLine(dealer.companyName);
    System.Console.Write("Contact Name: ");
    System.Console.WriteLine(dealer.contactName);
    System.Console.Write("Contact Phone: ");
    System.Console.WriteLine(dealer.contactPhone);
    System.Console.WriteLine("----------------------------------------");
    System.Console.WriteLine("Store Inventory");
    System.Console.Write("Item #:  ");
    System.Console.WriteLine(si.itemNumber);
    System.Console.Write("Item Name: ");
    System.Console.WriteLine(si.itemName);
    System.Console.Write("Unit Price: ");
    System.Console.WriteLine(si.unitPrice);
    System.Console.WriteLine("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=");
    System.Console.ReadKey();
    return 0;
  }
}
```

۲. برنامه را اجرا کنید، نتیجه ی زیر به دست می آید.

```
Manufacturer Information
Company Name: Peel Corp
Contact Name: Sylvain Yobo
Contact Phone: (602) 791-8074
----------------------------------------
Store Inventory
Item #:  613508
Item Name: Merino Crew Neck Cardigan
Unit Price: 80.00
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
```

۳. Enter را زده و به محیط برنامه نویسی باز گردید.

گنجاندن یک فضای نامی در دل فضای نامی دیگر

می توان یک فضای نام را داخل فضای نام دیگر قرار داد. به چنین فرایندی nesting a namespace یا تودرتو کردن فضای نام می گویند.

برای ایجاد فضای نام داخل فضای نام دیگر، از همان شیوه ای که برای ایجاد فضای نام ساده استفاده می کنید پیروی کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
namespace Business
{
  public class House
  {
    public string propertyNumber;
    public decimal price;
  }
  namespace Dealership
  {
  }
}
```

در مثال بالا، فضای نام Dealership داخل فضای نام Business گنجانده شده. پس از ایجاد فضاهای نام دلخواه، می توان به ساخت کلاس های مورد نیاز داخل فضای نام ذکر شده پرداخت.

به منظور دستیابی به اعضای فضای نام، پیش از فراخوانی عضو مورد نظر فضای نام (یا فضای نامی که در دل فضای نام دیگر گنجانده شده) از عملگر نقطه (.) استفاده می کنیم.

مثال

```
namespace Business
{
  public class House
  {
    public string propertyNumber;
    public decimal price;
  }
  namespace Dealership
  {
    public class Car
    {
      public decimal price;
    }
  }
}
public class Exercise
{
  static void Main()
  {
    Business.House property = new Business.House();
    property.propertyNumber = "D294FF";
    property.price = 425880;
    Business.Dealership.Car vehicle = new Business.Dealership.Car();
    vehicle.price = 38425.50M;
  }
}
```

به همین ترتیب می توان هر تعداد فضای نام که لازم می دانید داخل فضای نام دیگر قرار دهید.

شیوه ی دیگری که با آن می توان فضای نامی را در فضای نام دیگر دخیل کرد، این است که به ایجاد فضای نام جدید بپردازید. پس از اسم فضای نام، عملگر نقطه و به دنبال آن اسم فضای نام دوم (فضای نامی که داخل فضای نام دیگر قرار می گیرد) را تایپ کنید.

```
namespace Geometry.Quadrilaterals
{
}
```

پس از ایجاد فضای نام مورد نظر، شما می توانید با تعریف کردن فضای نام به محتوای آن دسترسی داشته باشید.

مثال

```
namespace Geometry.Quadrilaterals
{
  public class Square
  {
    public double side;
  }
}
public class Exercise
{
  public static void Main()
  {
    Geometry.Quadrilaterals.Square sqr = new Geometry.Quadrilaterals.Square();
    sqr.side = 25.85;
  }
}
```

به همین صورت شما می توانید چند NameSpace را داخل یک NameSpace مانند زیر بگنجانید:

```
namespace Geometry.Quadrilaterals
{
}

namespace Geometry.Rounds
{
}
```

به همین ترتیب، می توان هر تعداد فضای نام که مورد نیاز است داخل فضای نام دیگر ایجاد کرد. پس از گنجاندن فضای نام (در فضای نام دیگر) برای دسترسی داشتن به محتوای آن، می توانید اسم انتخابی آن را تعریف کنید.

مثال

```
namespace Geometry.Quadrilaterals
{
  public class Square
  {
    public double side;
  }
}
namespace Geometry.Volumes.Elliptic
{
  public class Cylinder
  {
    public double radius;
  }
}
public class Exercise
{
  public static void Main()
  {
    Geometry.Quadrilaterals.Square sqr = new Geometry.Quadrilaterals.Square();
    Geometry.Volumes.Elliptic.Cylinder cyl = new Geometry.Volumes.Elliptic.Cylinder();
    sqr.side = 25.85;
    cyl.radius = 36.85;
  }
}
```

گنجاندن فضای نام

۱. روی (لیبل) Records.cs کلیک کنید تا به فایل آن دست پیدا کرده و آن را به صورت زیر تغییر دهید.

```
namespace Store
{
  namespace Inventory
  {
    public class StoreItem
    {
      public int itemNumber;
      public string itemName;
      public decimal unitPrice;
    }
  }
  namespace Personel
  {
    namespace PayrollRecords
    {
      public class Employee
      {
        public string firstName;
        public string lastName;
```

```
        public decimal HourlySalary;
      }
      public class Contractors
      {
        public string FullName;
        public int ContractStatus;
      }
    }
  }
}
```

۲. حال فایل DepartmentStore.cs را باز کرده و آن را به ترتیب زیر اصلاح کنید.

```
using Supply;
 using Store.Inventory;
 public class DepartmentStore
 {
   static int Main()
   {
     . . . No Change
     return 0;
   }
 }
```

سربارگذاری کلاس امری امکان ناپذیر

برخلاف متدهای کلاس، نمی توان اسم کلاس را داخل فایل سربارگذاری (overload) کرد. به عبارت دیگر، نمی توان دو کلاس هم نام در یک محدوده (scope) ایجاد کرد. یکی از گزینه های پیش رو این است که هر فضای نامی را داخل کلاس خودش قرار دهید.

مثال

```
 namespace Arithmetic
 {
   public class Numbers
   {
     public int value;
   }
 }
 namespace Algebra
 {
   public class Numbers
   {
     public int value;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
  }
}
```

گزینه ی دیگری نیز وجود دارد که در مبحث generics مطرح می گردد. به عبارتی مختصر، می توان در یک فضای نام، دو کلاس داشت که یکی عمومی (generic) باشد و دیگری generic نباشد.

مثال

```
namespace Arithmetic
{
  public class Numbers
  {
    public int value;
  }
  public class Numbers<T>
  {
    public int value;
  }
}
```

استفاده از کلاس نام گذاری شده

۱. ابتدا، فایل Suppliers.cs را بازکنید، سپس آن را به صورت زیر اصلاح کنید.

```
namespace Supply
{
  public class SalesPerson
  {
    public string fullName;
    public string expertise;
    public string phoneNumber;
  }
}
namespace Manufacturers.Domestic
{
  public class Manufacturer
  {
    public string companyName;
    public string contactName;
    public string contactPhone;
  }
}
namespace Manufacturers.Foreign.Asia
{
  public class Manufacturer
  {
```

```
    public string country;
    public string companyName;
    public string contactName;
    public string contactPhone;
    public string webSite;
  }
}
```

۲. حال، به فایل DepartmentStore.cs مراجعه کرده و آن را به صورت زیر تغییر دهید.

```
using Supply;
using Manufacturers.Foreign.Asia;
using Store.Inventory;
public class DepartmentStore
{
  static int Main()
  {
    . . . No Change
    return 0;
  }
}
```

اسم مستعار (alias) فضای نام

دسترسی به فضای نامی که آن فضای نام خود در دل فضای نام های دیگر گنجانده شده بسیار دشوار و زمان بر است. برای حل این مسئله، می توان از نام مستعار یک فضای نام بهره جست (که در واقع میانبری برای دست یافتن به فضای نامی است که در دل فضای نامی دیگر جای گرفته). برای ایجاد نام مستعار، کلیدواژه ی using و به دنبال آن اسم دلخواه، علامت =، و namespace را تایپ کنید.

نحوه ی ایجاد و استفاده از فضای نام

۱. پس از باز کردن فایل DepartmentStore.cs، آن را به صورت زیر اصلاح کنید.

```
using Supply;
using Asian = Manufacturers.Foreign.Asia;
using Store.Inventory;
public class DepartmentStore
{
  static int Main()
  {
    Asian.Manufacturer dealer = new Asian.Manufacturer();
    dealer.companyName = "Peel Corp";
    dealer.contactName = "Sylvain Yobo";
    dealer.contactPhone = "(602) 791-8074";
    StoreItem si = new StoreItem();
    si.itemNumber = 613508;
```

```
    si.itemName = "Merino Crew Neck Cardigan";
    si.unitPrice = 80.00M;
    System.Console.WriteLine("Manufacturer Information");
    System.Console.Write("Company Name: ");
    System.Console.WriteLine(dealer.companyName);
    System.Console.Write("Contact Name: ");
    System.Console.WriteLine(dealer.contactName);
    System.Console.Write("Contact Phone: ");
    System.Console.WriteLine(dealer.contactPhone);
    System.Console.WriteLine("---------------------------------------");
    System.Console.WriteLine("Store Inventory");
    System.Console.Write("Item #:  ");
    System.Console.WriteLine(si.itemNumber);
    System.Console.Write("Item Name: ");
    System.Console.WriteLine(si.itemName);
    System.Console.Write("Unit Price: ");
    System.Console.WriteLine(si.unitPrice);
    System.Console.WriteLine("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=");
    System.Console.ReadKey();
    return 0;
  }
}
```

۲. برنامه را اجرا کنید.

۳. پنجره ی DOS را بسته و به محیط برنامه نویسی برگردید.

مدیریت فضای نام

درج کردن فضای نام

چنانچه کدی از قبل آماده دارید و می خواهید آن را داخل یک فضای نام قرار دهید.

روی بالاترین قسمت کد نویسی کلیک کرده، namespace و به دنبال آن یک اسم و علامت { را درج کنید. اکنون به انتهای بخش کد نویسی مراجعه کرده و کاراکتر { را تایپ کنید.

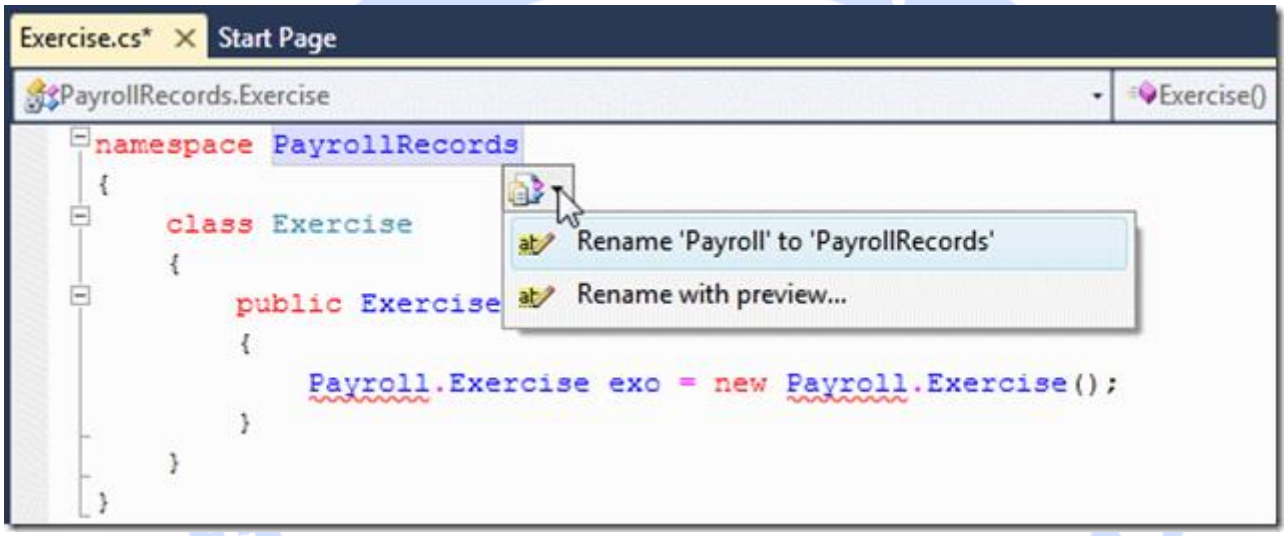
کدی را که می خواهید در فضای نام دخیل کنید انتخاب کرده، روی گزینه ی انتخابی کلیک کنید. سپس Surround With... را انتخاب کرده و در لیستی که ظاهر می شود دوبار روی namespace کلیک کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

تغییر اسم فضای نام

فرایند تغییر اسم فضای نام، از همان منطقی که برای تغییر اسم متغیرها،کلاس ها، و متدها تعریف کردیم پیروی می کند. اگر نمی خواهید خود به صورت دستی اسم را تغییر دهید، می توانید از Code Editor استفاده کنید. برای این منظور

در Code Editor، ابتدا اسم فضای نام مورد نظر را پیدا کنید. سپس روی پیکان تَگ فضای نام مذکور کلیک کرده و یک گزینه را از فهرست آن انتخاب کنید.



روی اسم راست کلیک کرده، سپس گزینه ی Rename... را انتخاب کنید.

معرفی فضای نام توکار (built-in namespaces)

پنجره ی Object Browser

برنامه ی.NET Framework کتابخانه ای بسیار غنی و گسترده است. مزیت برجسته ی آن، مجموعه ی بزرگ و گسترده ی کلاس های آن است. برای سازمان دهی این کلاس ها، برنامه ی.NET Framework فضاهای نامی متعددی ایجاد کرده و به کار می برد. از هر فضای نامی به وجود آمده به منظور ایجاد یک مجموعه کلاس (های) منحصر به فرد استفاده می شود.

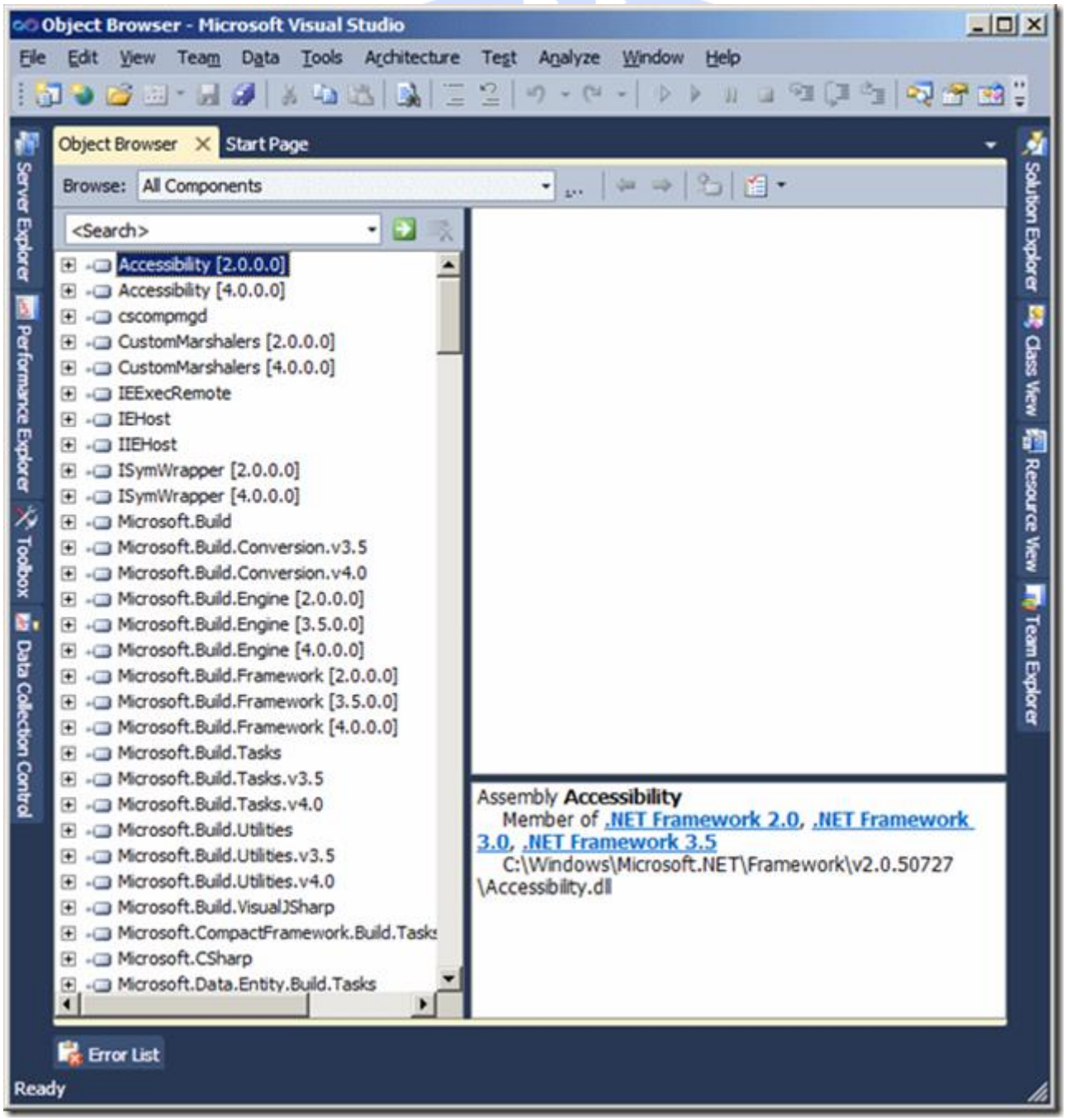
Microsoft Visual Studio پنجره ی Object Browser را برای بررسی و پویش فضاهای نامی برنامه ی.NET Framework ارائه می دهد. برای دسترسی به آن

فهرست گزینه ی اصلی را باز کرده، به View -> Object Browser مراجعه کنید.

در Standard toolbar روی دکمه ی Object Browser کلیک کنید.

اکنون، کلید F2 را فشار دهید.

Object Browser خود از سه پنجره تشکیل شده

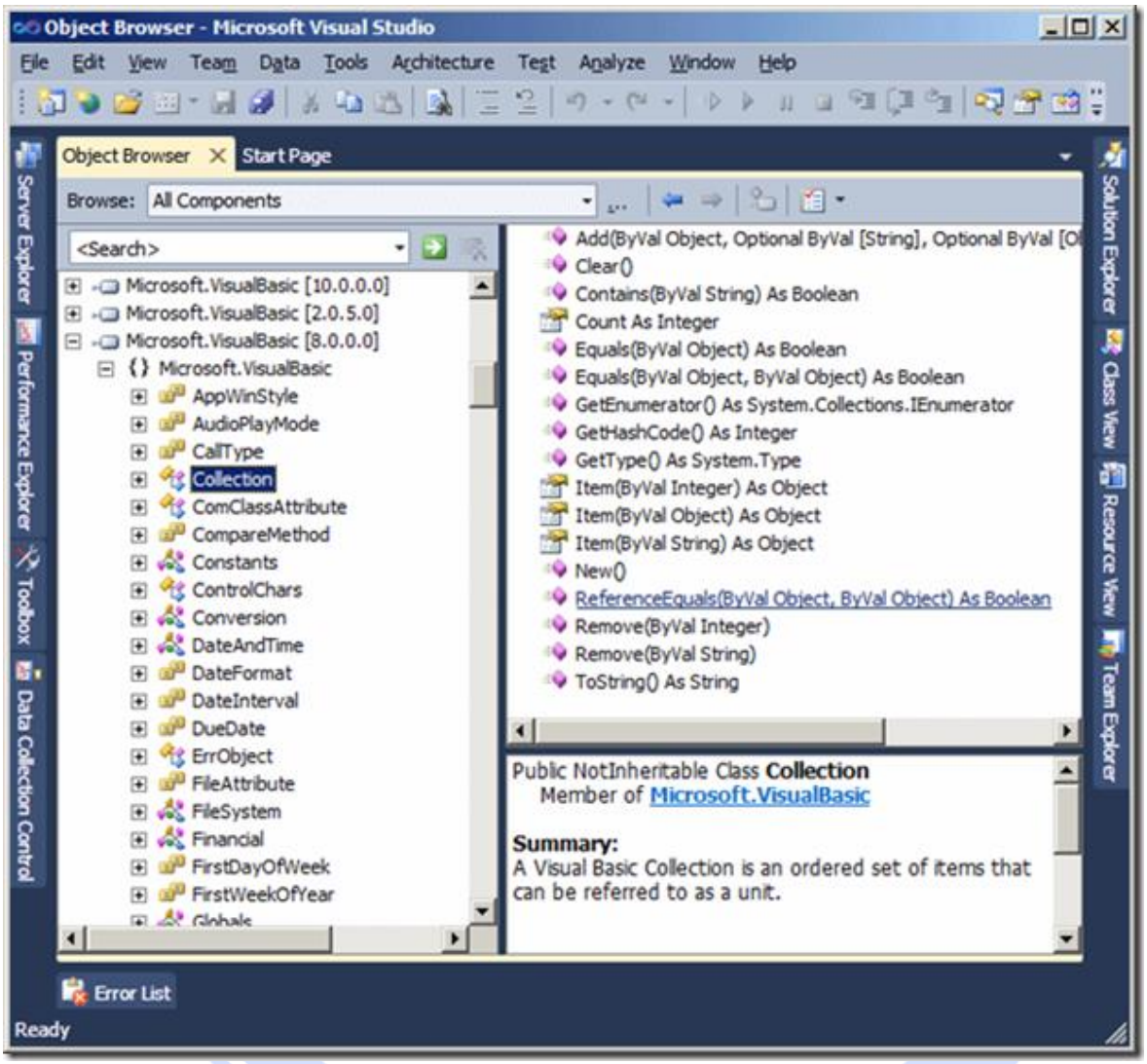


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

پنجره ی سمت چپ فهرستی از کتابخانه های موجود را نمایش می دهد. همان طور که در تصویر فوق مشاهده می کنید برخی از گره ها (nodes) اعدادی داخل علامت [] دارند. این اعداد نشانگر نسخه ی NET Framework. هستند. چنانچه گره های مختلف ولی هم نام دارای [] بودند، بدانید که کتابخانه ی مورد نظر در نسخه های متفاوت NET Framework. ایجاد و به روز رسانی شده. برای مثال، 4.0.0.0 نشانگر ورژن Microsoft.NET Framework 4.0 است.

به این خاطر که اغلب کتابخانه های مذکور فضاهای نام متعددی دارند، هر گره مجهز به دکمه ی + می باشد. برای باز کردن گره، کافی است روی + کلیک کنید. پس از باز کردن کتابخانه مورد نظر، فضاهای نامی داخل آن به صورت گره پدیدار می شوند. به دلیل این که اکثر فضاهای نام حاوی چندین (بیش از یک کلاس) کلاس هستند، هر فضای نام مجهز به دکمه ی + می باشد. همان طور که پیش تر ذکر شد برای باز کردن آن، باید روی + کلیک کنید. پس از باز کردن فضای نام دلخواه، فهرست کلاس های آن ظاهر می شود. برای مشاهده ی اعضای یک کلاس (در پنجره ی سمت چپ) روی اسم کلاس مورد نظر کلیک کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



برای مشاهده ی توصیفی از یک عضو کلاس (در پنجره ی بالایی سمت راست)، روی آن عضو کلیک کنید. حال، پنجره ی پایینی سمت راست اطلاعاتی در رابطه با آیتم انتخابی شما ارائه می دهد.

معرفی فضای نام System

پرکاربرد ترین فضای نام در .NET Framework، System نام دارد. برای استفاده از فضای نام System باید آن را داخل یک متد تایپ کنید.

مثال

```
public class Exercise
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
{
  static int Main()
  {
    System.Console.WriteLine("The wonderful world of C# programming");
    return 0;
  }
}
```

همچنین می توان پیش از System، کلید واژه ی global و عملگر :: را به کار برد.

```
public class Exercise
{
  static int Main()
  {
    global::System.Console.WriteLine("The wonderful world of C# programming");
    return 0;
  }
}
```

پیش تر ذکر شد که برای دستیابی به یک فضای نام، از کلید واژه ی using استفاده می شود. از همین روش برای اشاره و دسترسی به فضای نامی System نیز استفاده می شود.

مثال

```
using System;
public class Exercise
{
  static int Main()
  {
    System.Console.WriteLine("The wonderful world of C# programming");
    return 0;
  }
}
```

این بار نیز می توانید global:: را پیش از کلیدواژه ی System به کار ببرید.

```
using global::System
public class Exercise
{
  static int Main()
  {
    System.Console.WriteLine("The wonderful world of C# programming");
    return 0;
  }
}
```

پس ازاین که using System یا global::System را به کار بردید، می توانید کلید واژه ی System را در بدنه ی تابع (function) حذف کنید.

مثال

```
using System;
 public class Exercise
  {
    static int Main()
    {
      Console.WriteLine("The wonderful world of C# programming");
      return 0;
    }
  }
```

معرفی دیگر فضاهای نامی

.NET Framework لیست بلند بالایی از فضاهای نام فراهم می کند. در این مبحث به همگی آن ها نمی پردازیم، ولی در جایی که باید از کلاس ها استفاده کنیم، خواهیم گفت هر کلاس در کدام فضای نام قرار می گیرد.

می توان با انتخاب گزینه ی Add New Class از فهرست گزینه ی اصلی پنجره ی Solution Explorer یا Class View، کلاسی جدید ایجاد کرد. در صورت استفاده از روش بالا visual studio فضاهای نام دیگری نیز اضافه می کند. در حال حاضر، کد را همان گونه هست (بدون ایجاد تغییر) استفاده می کنیم.

نوع داده های.NET

تمامی نوع داده هایی که تا کنون به کار برده ایم در .NET Framework در قالب (توسط) کلاس ها نشان داده می شوند. به عبارت دیگر، نوع داده های مذکور خود مجهز به متد هستند. کلاس های مزبور در فضای نام System تعریف می شوند. کلاس های نوع داده های نام برده بدین ترتیب تعریف می شوند.

C# Data Type (نوع داده ی C#)	Equivalent.NET Class (کلاس معادل در.NET)	C# Data Type	Equivalent.NET Class
bool	Boolean	char	Char
byte	Byte	sbyte	SByte
short	Int16	ushort	UInt16
int	Int32	uint	UInt32
long	Int64	ulong	UInt64
float	Single	double	Double
decimal	Decimal		

به عبارت دیگر، چنانچه تمایلی به استفاده از نوع داده هایی که تا کنون معرفی کرده ایم ندارید، می توانید از کلاسی که در فضای نام System تعریف شده استفاده کنید. به منظور استفاده از کلاس های گفته شده، کافی است اسم آن را تایپ کنید.

مثال

```
class Operations
{
  public double Addition()
  {
    Double a;
    Double b;
    Double c;
    a = 128.76;
    b = 5044.52;
    c = a + b;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
        return c;
    }
}
```

به دلیل این که نوع داده ها به عنوان (در قالب) کلاس تعریف می شوند، خود مجهز به متد هستند. یکی از متدهایی که همگی آن ها دارند ToString نامیده می شود. همان طور که از اسم آن پیدا است، این متد به منظور تبدیل یک مقدار به رشته به کار می رود.

مقدمه ای بر کتابخانه های سفارشی

.NET Framework کتابخانه ای عظیم متشکل از کلاس های گوناگون و مختلف هست که هر یک برای منظور خاصی به کار می رود. درباره ی برخی از کلاس های نام برده در مباحث پیش رو بحث خواهیم کرد. با این وجود، چنانچه .NET Framework قابلیت مورد نیاز را فراهم نکرد، شما می توانید کتابخانه ی دلخواه خود را ایجاد کرده و آن را در چندین برنامه به کار ببرید. همچنین می توانید کتابخانه ای بسازید که جنبه ی تجاری داشته، و بتوان آن را توزیع کرده و به فروش رساند. برای مثال، می توان چندین کلاس آماده داخل یک کتابخانه گنجاند و در دسترس دیگران قرار داد.

کتابخانه برنامه ای است شامل کلاس ها / یا دیگر منابع مورد نیاز برنامه های دیگر. به منظور طراحی و ایجاد برنامه ی مزبور از همان روشی استفاده می کنیم که برای برنامه های دیگر (برنامه هایی که تاکنون و در دروس پیشین ایجاد کردیم) به کار بردیم. به این خاطر که کتابخانه یک فایل یا برنامه ی اجرایی نیست، نیازی هم به تابع Main() ندارد. لازم به ذکر است که کتابخانه به طور معمول پسوند .dll را به همراه دارد (البته کتابخانه هایی از نوع دیگر نیز در Microsoft Windows وجود دارد که پسوند آن ها .lib می باشد).

ایجاد کتابخانه ی سفارشی

۱. برنامه ی Microsoft Visual C# را اجرا کنید.

۲. برای ایجاد برنامه ی جدید، فهرست اصلی را باز کرده، File -> New Project... را انتخاب کنید.

۳. گزینه ی Windows را از لیست سمت چپ انتخاب کنید.

۴. حال در لیست سمت راست روی Empty Project کلیک کنید.

۵. اسم را به SaleRecord تغییر داده، سپس روی گزینه ی OK کلیک کنید.

۶. به منظور ذخیره ی پروژه مورد نظر، به Standard toolbar مراجعه کرده، اکنون روی دکمه ی  Save All کلیک کنید.

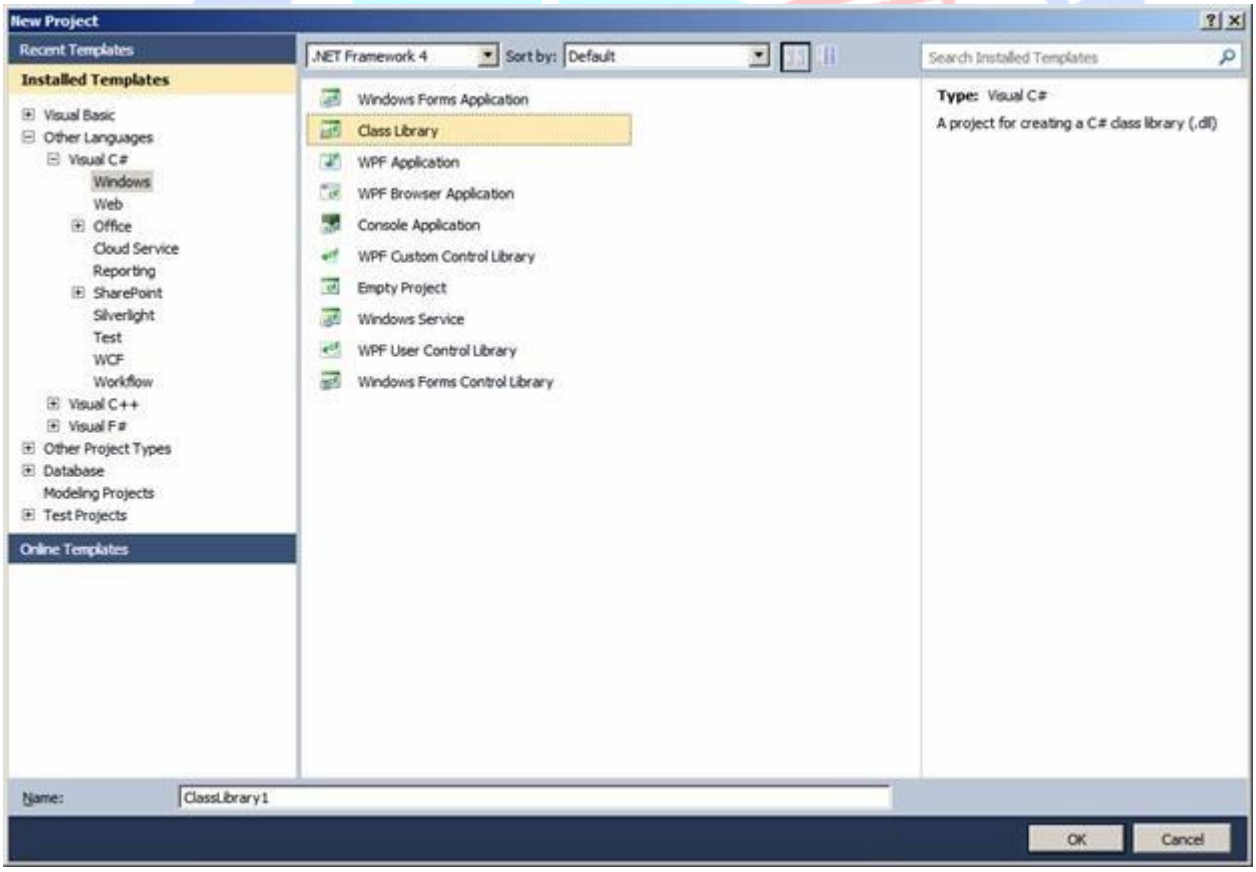
۷. در صورت پذیرفتن فولدر پیشنهادی (در مکانی که کامپیوتر توصیه می کند)، محل ذخیره سازی آن را به خاطر بسپرید. در غیر این صورت، مسیر دلخواه خود را وارد کنید و آن را به یاد بسپارید.

۸. گزینه ی Save را کلیک کنید.

نحوه ی ایجاد کتابخانه

کتابخانه ممکن است از تنها یک فایل تشکیل شده باشد یا از چندین فایل متعدد (هر تعداد که لازم است). فایلی که عضوی از یک کتابخانه هست می تواند یک یا چند کلاس دربرداشته باشد. هر کلاس باید کارکردی را پیاده و فراهم کند که سرانجام برای تمامی کلاس ها مفید واقع شده و در دسترس آن ها قرار گیرد. کلاس های یک کتابخانه درست مثل کلاس های دیگر برنامه ها ایجاد می شوند.

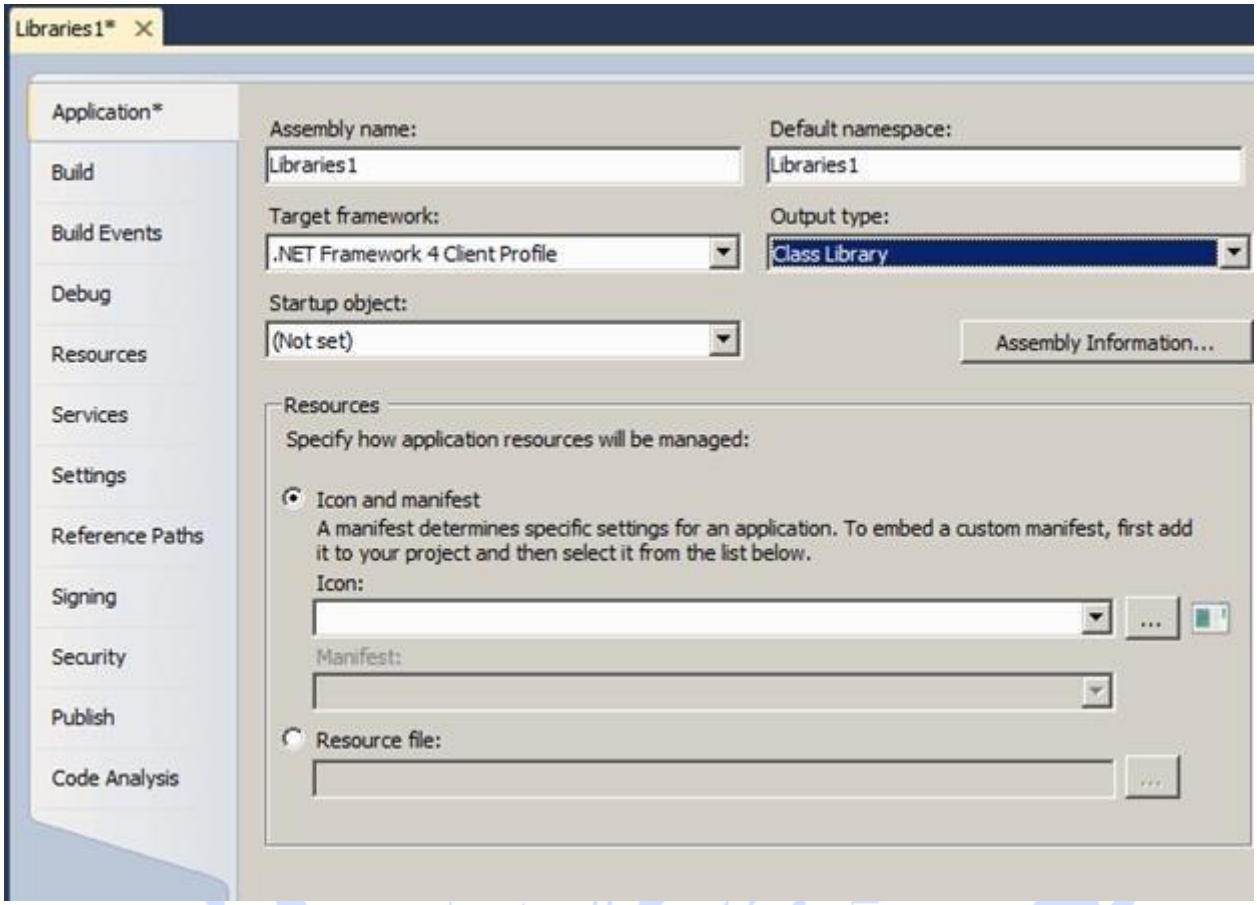
برای ایجاد کتابخانه، فهرست گزینه ی اصلی Microsoft Visual Studio را باز کرده، روی گزینه های File -> New Project.. کلیک کنید. در لیست میانی، می توانید روی Empty Project یا گزینه ی Class Library کلیک کنید. حال، اسمی برای آن انتخاب کنید.



در صورت انتخاب گزینه ی Empty Project در پنجره ی محاوره ی New Project، باید پنجره ی Properties را باز کرده، سپس در Output Type combo box گزینه ی Class Library را انتخاب کنید.

در هر دو صورت، skeleton code ای در اختیار شما قرار می گیرد که می توانید باب میلتان به کاربرید.

مثال

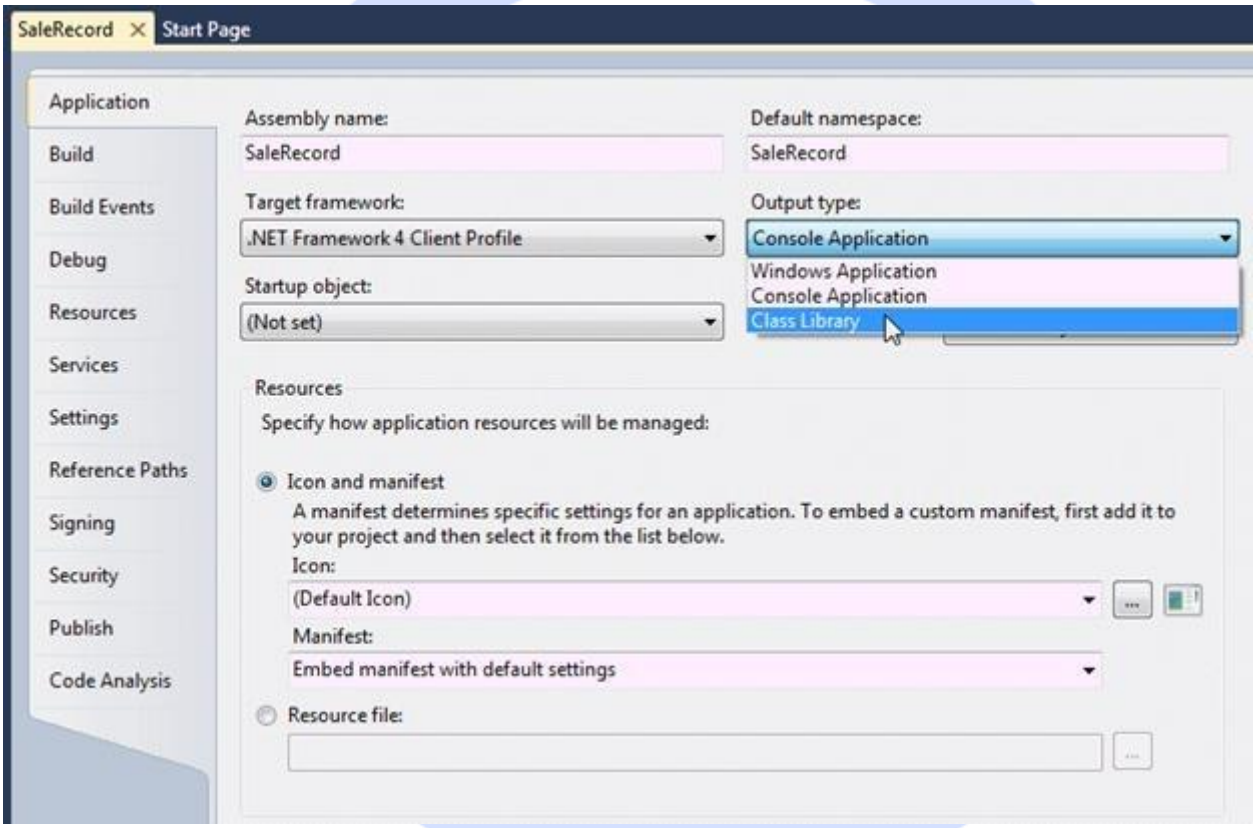



```
public class Algebra
{
  public double Addition(double x = 0, double y = 0)
  {
    return x + y;
  }
  public double Subtraction(double x = 0, double y = 0)
  {
    return x - y;
  }
  public double Multiplication(double x = 0, double y = 0)
  {
    return x * y;
  }
  public double Division(double x = 0, double y = 1)
  {
    return x / y;
  }
}
```

راه اندازی کتابخانه

۱. به فهرست گزینه ی اصلی (main menu) مراجعه کرده، روی Project -> SaleRecord Properties... کلیک کنید.

۲. در Output Type combo box گزینه ی Class Library را انتخاب کنید.



۳. برای ایجاد فایل جدید، در فهرست گزینه ی اصلی، روی گزینه های Project -> Add New Item... کلیک کنید.

۴. در لیست سمت چپ، Code را کلیک کنید.

۵. در لیست سمت راست، Code File را انتخاب کنید.

۶. اسم را به StoreItem تغییر داده، سپس روی Add کلیک کنید.

۷. فایل را به صورت زیر اصلاح کنید.

```
public class StoreItem
{
  public int itemNumber;
```

```
    public string itemName;
    public string size;
    public decimal unitPrice;
}
```

۸. تمامی تغییرات وارد آمده را با کلیک روی دکمه ی Save all ذخیره کنید.

ساختن کتابخانه

به این خاطر که فایل انتخابی کتابخانه هست نه یک برنامه / فایل اجرایی، برای ترجمه (compile) آن

در فهرست اصلی، می توانید روی Build -> Build *ProjectName* کلیک کنید.

در پنجره ی Solution Explorer، ابتدا روی اسم پروژه ی مورد نظر راست کلیک کرده، سپس گزینه ی Build را انتخاب کنید.

در پنجره ی Class View، روی اسم پروژه راست کلیک کرده، Build را انتخاب کنید.

برای ترجمه ی (compile) کتابخانه در پنجره ی فرمان (Command Prompt)، دستور زیر را تایپ کنید.

```
csc /target:library NameOfFile.cs
```

اکنون کلید Enter را بزنید.

ایجاد کتاب خانه در برنامه

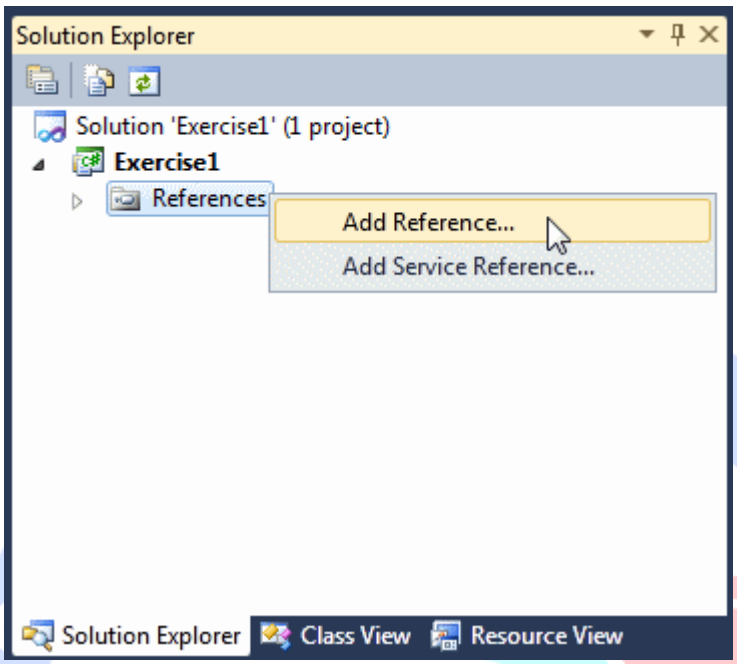
برای ایجاد کتابخانه، در فهرست اصلی روی گزینه های Build -> Build Solution کلیک کنید.

به کاربردن کتابخانه ی سفارشی (custom library)

پس از ایجاد پروژه می توانید آن (پروژه ی ایجاد شده) را در همان پروژه ای که کتابخانه در آن ساخته شده به کاربرید یا آن را در پروژه ای دیگر مورد استفاده ی خود قرار دهید، در صورت کار با Microsoft Visual Studio، کار خود را با ایجاد پروژه ی جدید آغاز کنید، به منظور استفاده از کتابخانه، در وهله ی اول باید به آن ارجاع (reference) دهید، برای این منظور

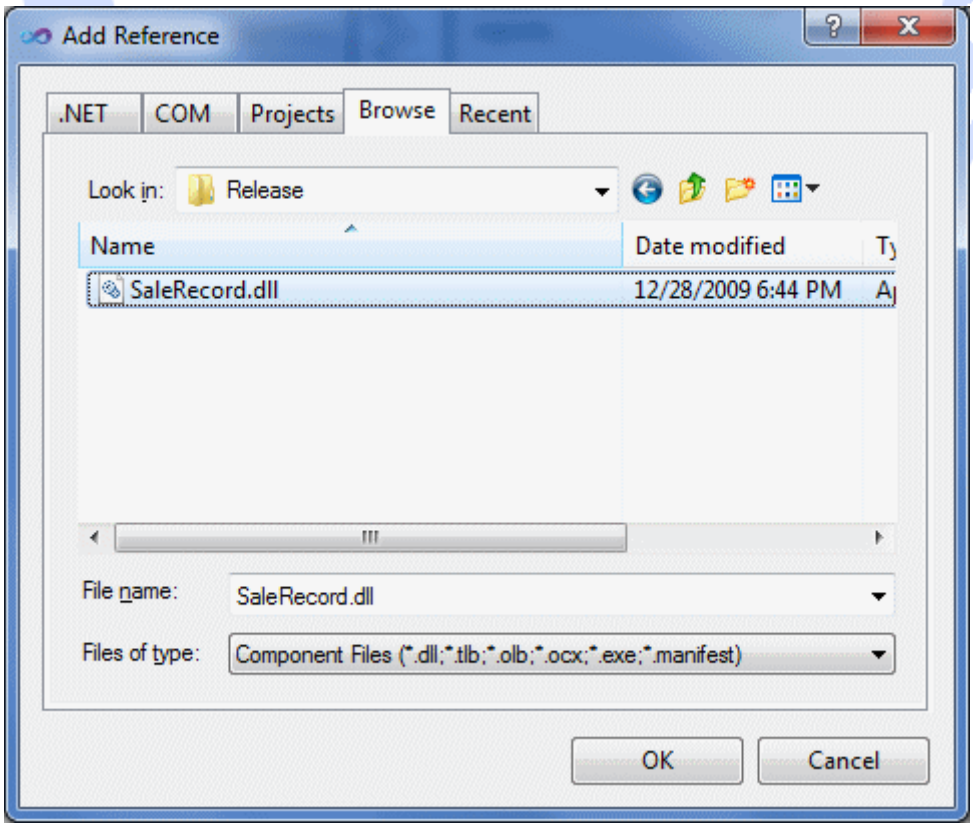
به فهرست گزینه ی اصلی مراجعه کرده، روی Project -> Add Reference... کلیک کنید.

در پنجره ی Solution Explorer، پس از راست کلیک روی References، گزینه ی Add Reference... را انتخاب کنید.



کلیک کرده، Browse پدیدار می گردد، حال می توانید روی تَبِ Add Reference در صورت انجام هر یک از دستورات بالا، پنجره ی محاوره

پس از پیدا کردن فولدری که کتابخانه در آن قرار دارد، روی فولدر بیان شده کلیک کنید.



آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

پس از انتخاب کتابخانه مورد نظر، OK را کلیک کنید، اکنون می توانید متدها و کلاس های کتابخانه را (به همان ترتیبی که از متدها و کلاس های کتابخانه ی NET Framework استفاده می کردید) به کار ببرید.

مثال

```
using System;
public class Exercise
{
  static void Main()
  {
    Algebra alg = new Algebra();
    double number1 = 244.58;
    double number2 = 5082.88;
    double result = alg.Addition(number1, number2);
    Console.Write(number1);
    Console.Write(" + ");
    Console.Write(number2);
    Console.Write(" = ");
    Console.WriteLine(result);
    System.Console.ReadKey();
  }
}
```

اگر می خواهید پروژه را در پنجره ی فرمان (Command Prompt) ترجمه (compile) کنید، دستوری مشابه به دستور زیر را تایپ کنید.

```
csc /target:library /out:DesiredNameOfLibrary.dll NameOfFile.cs
```

نتیجه ی زیر را به دست می دهد.

```
244.58 + 5082.88 = 5327.46
Press any key to continue...
```

استفاده از کتابخانه ی سفارشی

۱. برای راه اندازی پروژه ی جدید، در فهرست اصلی روی File -> New Project... کلیک کنید.

۲. در لیست میانی، گزینه ی Empty Project را انتخاب کنید.

۳. اسم را به DepartmentStore5 تغییر داده، سپس Add را کلیک کنید.

۴. در پنجره ی Solution Explorer، پایین DepartmentStore5، روی References راست کلیک کرده، سپس گزینه ی Add Reference... را از لیست انتخاب کنید.

۵. اکنون به تَبِ Browse مراجعه کنید.

۶. فولدری را که کتابخانه در آن ایجاد شده، پیدا کنید.

۷. فایل SaleRecord.dll را انتخاب کنید.

۸. OK را کلیک کنید.

۹. در پنجره ی Solution Explorer، روی DepartmentStore5 راست کلیک کرده، سپس DepartmentStore5 -> Add -> New Item...

۱۰. در لیست میانی روی Code File کلیک کنید.

۱۱. اسم را به DepartmentStore تغییر داده، سپس گزینه ی Add را انتخاب کنید.

۱۲. فایل را به صورت زیر اصلاح کنید.

```
using System;
public class Program
{
  static int Main()
  {
    StoreItem si = new StoreItem();
    si.itemNumber = 660284;
    si.itemName = "Tropical Wool Neutral Jacket";
    si.unitPrice = 200.00M;
    Console.WriteLine("Store Inventory");
    Console.Write("Item #:  ");
    Console.WriteLine(si.itemNumber);
    Console.Write("Item Name: ");
    Console.WriteLine(si.itemName);
    Console.Write("Unit Price: ");
    Console.WriteLine(si.unitPrice);
    Console.WriteLine("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=");
    System.Console.ReadKey();
    return 0;
  }
}
```

۱۳. حال برنامه را اجرا کنید.

```
Store Inventory
Item #:  660284
Item Name: Tropical Wool Neutral Jacket
```

```
Unit Price: 200.00
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Press any key to continue...
```

۱۴. به محیط برنامه نویسی باز گردید.

مقدمه ای بر کتابخانه های توکار

مقدمه ای بر کتابخانه ی C#

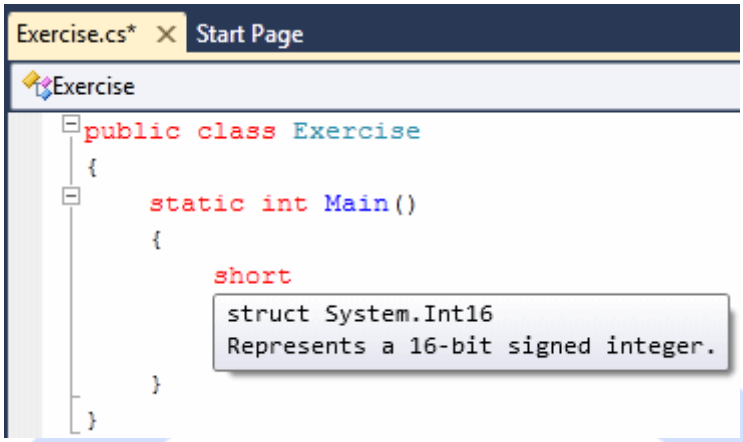
برخلاف خیلی از زبان های برنامه نویسی دیگر مثل : C++ یا Pascal (Delphi) (Object)، زبان C# فاقد کتابخانه ی توکار (متعلق به خود) می باشد. لازم به ذکر است که این فقدان به معنای کارایی و قابلیت کمتر و ضعف برنامه C# نیست. بلکه زبان C# به دلیل همین انعطاف پذیری بالایی که دارد، می تواند از کتابخانه های دیگر زبان های برنامه نویسی نهایت استفاده را ببرد.

در حقیقت C# فقط یک کتابخانه ی کوچک دارد متشکل از تنها چند کلاس ناچیز که اغلب به کار نمی اید. اسم این کتابخانه Microsoft.CSharp.dll می باشد. البته در نسخه ی تازه منتشر شده ی C# (C# 4.0)، کتابخانه به روزرسانی شده و نوع داده ی جدیدی به آن اضافه شده که کاربرد فراوان دارد.

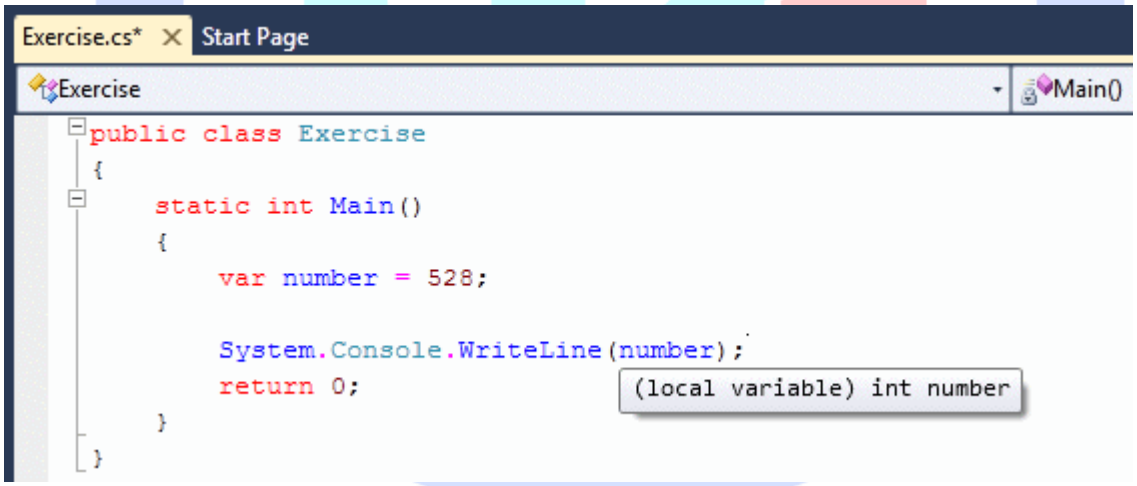
اگر برنامه ی کاربردی کنسول (console application) ایجاد کنید، Microsoft Visual Studio به صورت خودکار کتابخانه ی Microsoft.CSharp.dll را به برنامه ی شما اضافه می کند. حال اگر پروژه ی خالی ایجاد کرده اید ولی در هر صورت می خواهید از این کتابخانه استفاده کنید، کافی است با مراجعه به پنجره ی محاوره Add Reference، کتابخانه ی بیان شده را به پروژه ی خود اضافه کنید.

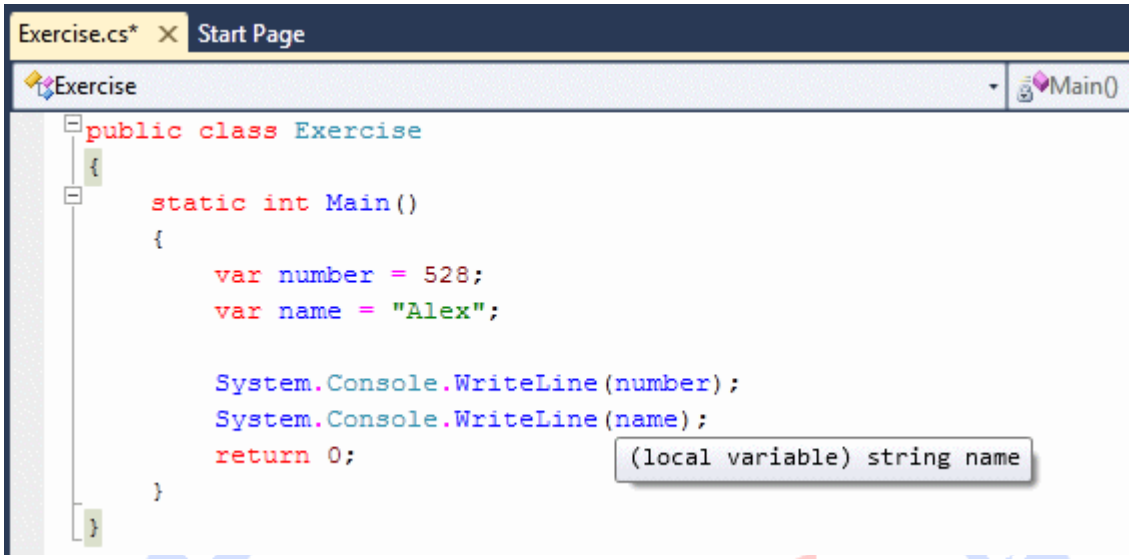
نوع داده ی پویا (dynamic data type)

در دروس پیشین، با نوع داده های مختلفی از short، int، float، double، decimal و string آشنا شدیم. گفته شد که نوع داده ی انتخابی به compiler خبر می دهد چه مقدار حافظه برای متغیر مورد نظر تخصیص یابد. در واقع، چنانچه (در Code Editor) نشانگر موس را روی نوع داده مورد نظر قرار دهید، راهنمای ابزار (tool tip) ای نمایان می شود که مقدار حافظه ی اشغال شده توسط متغیر را نشان می دهد.



این هم گفته شد که اگر تمایل ندارید نوع داده ای، هنگام تعریف متغیر، مشخص کنید می توانید ازکلیدواژه ی var استفاده کنید اما باید متغیر را پیش از به کاربردن مقداردهی اولیه کنید. پس از مقداردهی اولیه ی متغیر، compiler بر مبنای برآوردش از مقداری که به متغیر اختصاص داده اید، حافظه رزرو می کند. هنگام استفاده از متغیر، چنانچه نشانگر موس را روی آن قرار دهید، Code Editor نوع داده ای که به آن تخصیص داده اید را نشان می دهد.





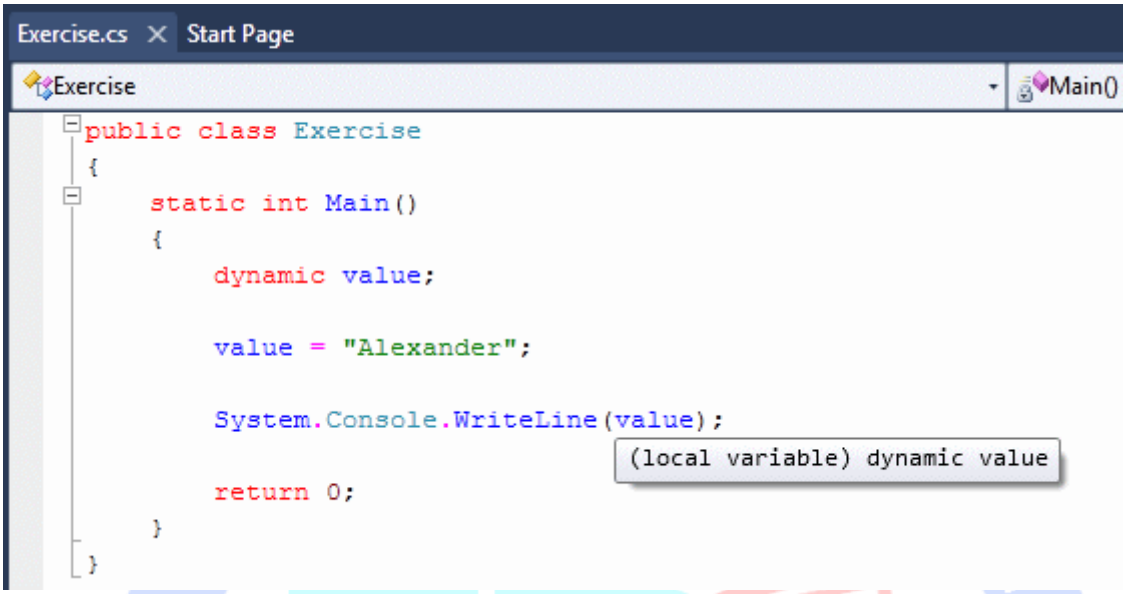
در برخی حالات، شما از مقدار فضای موردنیاز متغیر (تا زمانی که به آن نیاز پیدا نکرده و از آن استفاده نکرده اید) مطلع نیستید. به عبارت دیگر، compiler را با مسائلی همچون تدارکات متغیر (مثل اینکه به چه قدر حافظه نیاز دارد) درگیر نمی کنیم تا زمانی که عملیاتی که به متغیر نیاز دارد به آن (متغیر) دسترسی پیدا کند. به چنین متغیری، متغیر پویا یا داینامیک گفته می شود.

برای تعریف متغیر پویا، از کلید واژه ی dynamic استفاده می کنیم.

مثال

```
public class Exercise
{
  static int Main()
  {
    dynamic value;
    return 0;
  }
}
```

برخلاف متغیری که با کلیدواژه ی var تعریف می شود، لازم نیست متغیر پویا را حین تعریف (آن) مقداردهی اولیه کنید. به هر حال، باید مقداری به آن اختصاص دهید (ترجیحاً پیش از به کار بردن آن). هنگامی که مقداری به متغیر پویا اختصاص می دهید، compiler بررسی نمی کند چه مقدار حافظه مناسب یا مورد نیاز متغیر مزبور است. چنانچه نشانگر موس را روی متغیر قرار دهید، Code Editor تنها نوع متغیر (پویا) را به شما نشان می دهد :



متغیر پویا اغلب زمانی استفاده می شود که پروژه ی شما نیاز به برقراری ارتباط با برنامه ی کاربردی (application) خارجی دارد و از مقادیر آن برنامه استفاده می کند. نمونه ی آن Microsoft Word document یا صفحه گسترده ی Microsoft Excel است که توسط زبان C# طراحی شده، و کتابخانه ی خارجی وارد می کنند.

به کاربردن متغیر پویا

۱. برای ایجاد پروژه ی جدید، در فهرست اصلی روی File -> New Project... کلیک کنید.

۲. در لیست میانی، روی Empty Project کلیک کنید.

۳. اسم آن را به DepartmentStore7 تغییر داده، سپس روی Add کلیک کنید.

۴. در پنجره ی Solution Explorer، روی DepartmentStore7 راست کلیک کرده سپس DepartmentStore7 -> Add -> New Item... را کلیک کنید.

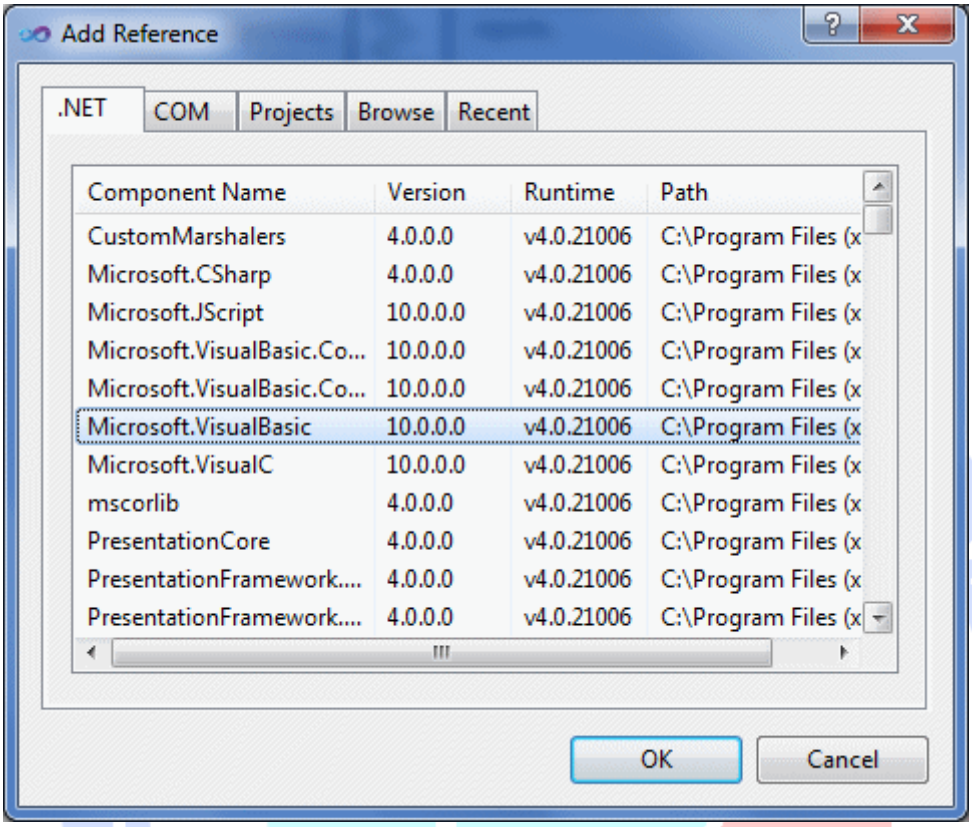
۵. گزینه ی Code File را از لیست میانی انتخاب کنید.

۶. اسم را به PayrollProcessing تغییر داده و روی Add کلیک کنید.

۷. به فهرست گزینه ی اصلی مراجعه کرده و گزینه های Project -> Add Reference... را انتخاب کنید.

۸. حال روی تَب .NET کلیک کنید.

۹. در لیست موردنظر، Microsoft. CSharp را انتخاب کنید.



۱۰. OK را کلیک کنید.

۱۱. فایل را به صورت زیر اصلاح کنید.

```
using System;
public class Payroll
{
   static int Main()
   {
      dynamic employeeName;
      dynamic hourlySalary, weeklyTime, weeklySalary;
      employeeName = "Patricia Katts";
      hourlySalary = 22.75;
      weeklyTime = 38.50;
      weeklySalary = hourlySalary * weeklyTime;
      Console.WriteLine("==================================");
      Console.WriteLine("Payroll Summary");
      Console.WriteLine("------------------------------");
      Console.Write("Employee Name: ");
      Console.WriteLine(employeeName);
```

```
        Console.Write("Hourly Salary: ");
        Console.WriteLine(hourlySalary);
        Console.Write("Weekly Time:  ");
        Console.WriteLine(weeklyTime);
        Console.Write("Weekly Salary: ");
        Console.WriteLine(weeklySalary);
        Console.WriteLine("==============================");
        System.Console.ReadKey();
        return 0;
    }
}
```

۱۲. برنامه را اجرا کنید. نتیجه ی زیر حاصل می گردد.

```
==============================
Payroll Summary
------------------------------
Employee Name: Patricia Katts
Hourly Salary: 22.75
Weekly Time:  38.5
Weekly Salary: 875.875
==============================
```

۱۳. پنجره ی DOS را بسته و به محیط برنامه نویسی بازگردید.

قابلیت همکاری

یکی از اهداف اصلی که در .NET دنبال می شود، فراهم کردن امکان همکاری بین زبان های مختلف است (مانند به اشتراک گذاری کد). یکی از روش هایی که این امر را امکان پذیر می کند، توانایی استفاده از قابلیت های یک زبان در برنامه ای است که توسط زبان برنامه نویسی دیگر نوشته شده است.

به طور مثال، می توان از (کتابخانه ی غنی) توابع Visual Basic در یک برنامه ی کاربردی C# بهره جست.

به این خاطر که هیچ کتابخانه ای صد در صد کامل نیست، ممکن است به قابلیتی نیاز پیدا کنید که در خود زبان انتخابی (زبانی که هم اکنون برای برنامه نویسی به کار می برید) موجود نباشد.

همچنین ممکن است با تیم برنامه نویسی همکاری داشته باشید که از زبان های برنامه نویسی مختلفی استفاده کرده و توابع یا عملیات پیچیده ای نوشته باشند که شما مجبور به استفاده در برنامه ی خود هستید.

کتابخانه ی win32

اصل سیستم عامل مایکروسافت ویندوز در محیط زبان C نوشته شده است (زبان والد C++، C# وJava).

برای ایجاد امکان برنامه نویسی (ایجاد برنامه های کاربردی)، مایکروسافت کتابخانه ای به نام Win32 را منتشر کرد. کتابخانه ی نام برده متشکل از یک سری تابع و کلاس هست که از قبل استفاده می کردید.

امروزه، شما دیگر برای ایجاد برنامه ی کاربردی ویندوز ملزوم به استفاده از Win32 نیستید.

با این وجود، هنوز شمار زیادی از عملیاتی که ملزم با انجام آن ها در برنامه کاربردی ویندوز هستید تنها در Win32 موجود هستند. خوشبختانه، به کاربردن تعداد زیادی از این توابع در زبان C# دشوار نیست و تنها قوانینی دارد که باید رعایت کرد.

مثال

```
using System;
using System.Runtime.InteropServices;
class Program
{
  [DllImport("Kernel32.dll")]
  public static extern bool SetConsoleTitle(string strMessage);
  static void Main()
  {
    SetConsoleTitle("C# Programming");
  }
}
```

کتابخانه ی CLI/Visual C++

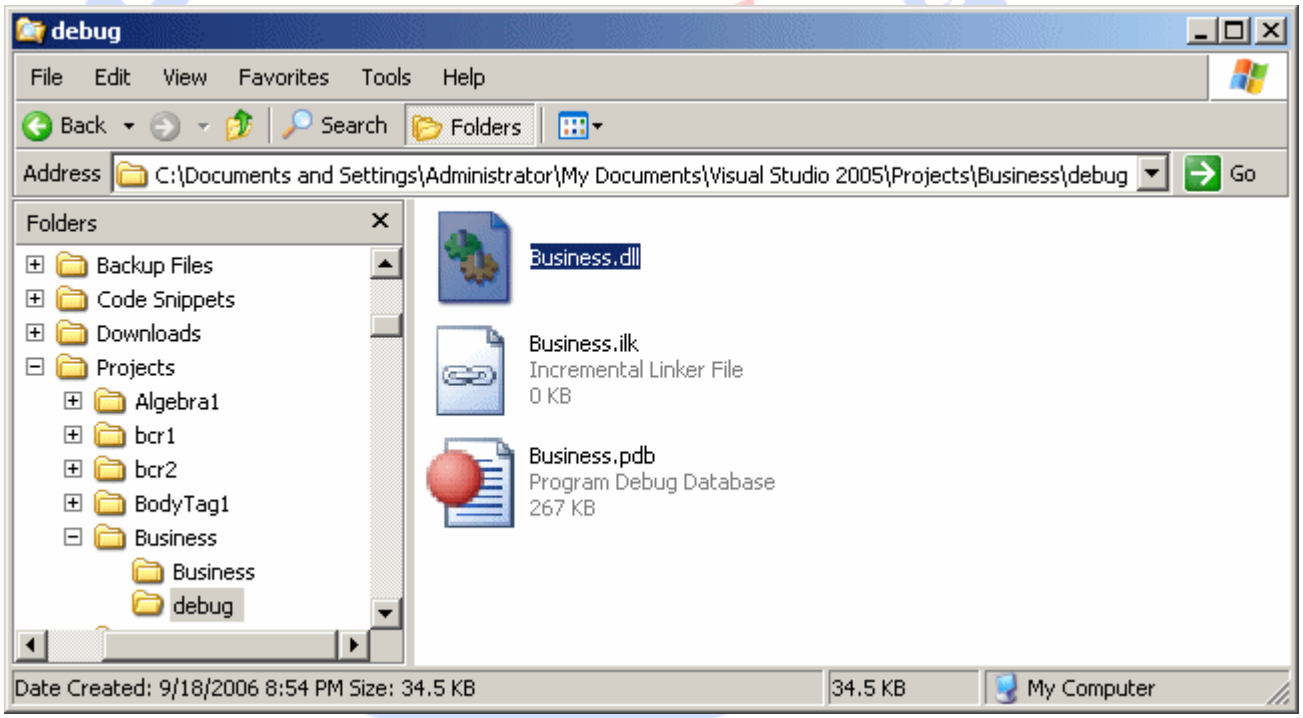
در زمان های قبل ایجاد یک کتابخانه (به خصوص در C++) کار بسیار دشواری بود. خوشبختانه، برنامه ی Microsoft Visual C++ اکنون ایجاد کتابخانه را به فرایندی فوق العاده ساده تبدیل کرده. برای این منظور، ابتدا باید پنجره ی محاوره ای New Project را باز کنید. پس از انتخاب Visual C++، در لیست میانی، روی گزینه ی Class Library کلیک کرده و اسم آن را مشخص کنید. حال می توانید در بدنه ی فایل، کلاس ها / توابع دلخواه را ایجاد کنید.

مثال

```
// Business.h
```

```
#pragma once
using namespace System;
public ref class Finance
{
public:
  double CalculateDiscount(double MarkedPrice,
           double DiscountRate)
  {
return MarkedPrice * DiscountRate / 100;
  }
};
```

پس از آماده شدن پروژه، باید به ساختن آن پرداخت (فهرست گزینه ی اصلی، Build -> Build Business). در نتیجه، compiler فایلی با پسوند .dll ایجاد می کند.



به کاربردن کتابخانه

ایجاد کتابخانه در محیط C++ بسیار آسان است. برای استفاده از آن، قواعدی هست که باید رعایت شود. در وهله ی اول، باید مطمئن شوید که پروژه ی شما می تواند کتابخانه ی موردنظر را پیدا کند. آسان ترین روش برای این منظور، کپی کردن فایل dll و جای گذاری آن در فولدر دربردارنده ی فایل اجرایی (executable) پروژه است. این کار را می توان به طور مستقیم (از طریق Microsoft Visual Studio) با وارد کردن فایل کتابخانه نیز انجام داد.

لازم است فضای نام System.Runtime.InteropServices را در پروژه ی خود دخیل کنید. پیش از کتابخانه (پیش از ("Business.dll")، صفت مشخصه ی DllImport را وارد کنید. این ویژگی (Attribute) اسم کتابخانه را که در قالب رشته (string) ارسال شده، به جای / به عنوان آرگومان می گیرد.

مثال

```
using System;
using System.Runtime.InteropServices;
using Business;
class Exercise
{
  [DllImport("Business.dll")]
  public static extern double CalculateDiscount(double price, double discount);
  static int Main()
  {
    Finance fin = new Finance();
    double markedPrice = 275.50;
    double discountRate = 25.00; // %
    double discountAmount = fin.CalculateDiscount(markedPrice,discountDate);
    double netPrice = markedPrice - discountAmount);
    Console.Write("Marked Price:  ");
    Console.WriteLine("markedPrice");
    Console.Write("Discount Rate:  ");
    Console.WriteLine("discountRate / 100");
    Console.Write("Discount Amount: ");
    Console.WriteLine("discountAmount");
    Console.Write("Net Price:    ");
    Console.WriteLine("netPrice");
    return 0;
  }
}
```

استاتیک

تصور کنید کلاسی به وجود آورده اید به نام Book. برای دسترسی به کلاس Book در متد Main()، باید در وهله ی اول متغیر آن را تعریف کنید. متغیر تعریف شده ی کلاس، نمونه ای از آن کلاس نیز اطلاق می گردد. به همین شکل می توان هر تعداد نمونه که مورد نیاز است از کلاس (واحد) تعریف کرد.

```
public class Book
{
  public string title;
  public string author;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    public short yearPublished;
    public int numberOfPages;
    public char coverType;
  }
  public class Exercise
  {
    static int Main()
    {
      var written = new Book();
      var bought = new Book();
      return 0;
    }
  }
```

هر یک از نمونه های گفته شده، دسترسی به عضوهای کلاس مربوط را فراهم می کند. ولی باید در نظر داشت که هر نمونه تنها مقدارهای خاص اعضای نمونه ی مختص به خود را نگه می دارد. توجه خود را به مثال زیر جلب کنید.

```
public class Exercise
  {
    static int Main()
    {
      var first = new Book();
      first.title = "Psychology and Human Evolution";
      first.author = "Jeannot Lamm";
      first.yearPublished = 1996;
      first.numberOfPages = 872;
      first.coverType = 'H';
      Console.WriteLine("Book Characteristics");
      Console.Write("Title: ");
      Console.WriteLine(first.title);
      Console.Write("Author: ");
      Console.WriteLine(first.author);
      Console.Write("Year:  ");
      Console.WriteLine(first.yearPublished);
      Console.Write("Pages: ");
      Console.WriteLine(first.numberOfPages);
      Console.Write("Cover: ");
      Console.WriteLine(first.coverType);
      var second = new Book();
      second.title = "C# First Step";
      second.author = "Alexandra Nyango";
      second.yearPublished = 2004;
      second.numberOfPages = 604;
      second.coverType = 'P';
      Console.WriteLine("Book Characteristics");
      Console.Write("Title: ");
      Console.WriteLine(second.title);
      Console.Write("Author: ");
      Console.WriteLine(second.author);
```

```
        Console.Write("Year: ");
        Console.WriteLine(second.yearPublished);
        Console.Write("Pages: ");
        Console.WriteLine(second.numberOfPages);
        Console.Write("Cover: ");
        Console.WriteLine(second.coverType);
        Console.ReadKey();
        return 0;
    }
}
```

نتیجه ی زیر حاصل می گردد.

```
Book Characteristics
Title: Psychology and Human Evolution
Author: Jeannot Lamm
Year: 1996
Pages: 872
Cover: H
Book Characteristics
Title: C# First Step
Author: Alexandra Nyango
Year: 2004
Pages: 604
Cover: P
```

کلیه ی متغیرهای عضو و متدهای کلاس که تاکنون به کار برده ایم تحت عنوان متغیر نمونه یا instance variable تعریف می شوند زیرا، به منظور دسترسی به آن ها ملزوم به تعریف نمونه ی کلاس در کلاسی دیگر هستید (منظور کلاسی است که می خواهید در آن به متغیر نمونه دست پیدا کنید).

می توانید متغیری در برنامه ی کاربردی خود، صرفنظر از اینکه کدام نمونه از (یک) شی را برای این منظور استفاده می کنید، تعریف کرده و به آن ارجاع دهید. چنین متغیری Static یا ایستا تلقی می گردد.

تعریف متغیر ایستا

برای تعریف متغیر (عضو یک) کلاس به عنوان ایستا، باید کلید واژه ی static را در سمت چپ آن تایپ کنید. به مثال زیر توجه کنید.

```
public class Book
{
    static string title;
}
```

به همین ترتیب می توانید هر تعداد متغیر ایستا که لازم می دانید معرفی کنید. همان طور که در مباحث پیشین تشریح شد، برای کنترل سطح دسترسی به فیلد های کلاس از modifier (تنظیم کننده های سطح دسترسی) استفاده می شود. در صورت استفاده از modifier، می توانید کلیدواژه ی static را پیش یا پس از آن به کار ببردید (البته مادام اینکه قبل از نوع داده استفاده شود).

مثال

```
public class Book
{
  public static string title;
  static public string author;
}
```

برای دسترسی به متغیر ایستا، باید ابتدا مکان به کار گیری آن را تعریف کنید (این که کجا می خواهید از آن استفاده کنید). این کار مستلزم تعیین (کردن) کلاس متغیر ایستا است.

مثال

```
public class Book
{
  public static string title;
  static public string author;
  public short yearPublished;
  public int pages;
  public char coverType;
}
public class Exercise
{
  static int Main()
  {
    var first = new Book();
    Book.title = "Psychology and Human Evolution";
    Book.author = "Jeannot Lamm";
    first.yearPublished = 1996;
    first.pages = 872;
    first.coverType = 'H';
    Console.WriteLine("Book Characteristics");
    Console.Write("Title: ");
    Console.WriteLine(Book.title);
    Console.Write("Author: ");
    Console.WriteLine(Book.author);
    Console.Write("Year:  ");
    Console.WriteLine(first.yearPublished);
    Console.Write("Pages: ");
```

```
        Console.WriteLine(first.Pages);
        Console.Write("Cover: ");
        Console.WriteLine(first.coverType);
        var second = new Book();
        Book.title = "C# First Step";
        Book.author = "Alexandra Nyango";
        second.yearPublished = 2004;
        second.pages = 604;
        second.coverType = 'P';
        Console.WriteLine("Book Characteristics");
        Console.Write("Title: ");
        Console.WriteLine(Book.title);
        Console.Write("Author: ");
        Console.WriteLine(Book.author);
        Console.Write("Year: ");
        Console.WriteLine(second.yearPublished);
        Console.Write("Pages: ");
        Console.WriteLine(second.Pages);
        Console.Write("Cover: ");
        Console.WriteLine(second.coverType);
        Console.ReadKey();
        return 0;
    }
  }
```

همان طور که در مثال بالا مشاهده می کنید زمانی که متغیری به عنوان ایستا تعریف می شود، دیگر نیازی به نمونه ی کلاس برای دسترسی به آن متغیر (عضو) از بیرون کلاس نیست. بر این اساس چنانچه همه ی اعضای کلاس را به عنوان ایستا معرفی کنید، دیگر لازم نیست برای دسترسی به آن ها متغیری از کلاسشان تعریف کنید. در مثال ذیل، فلیدهای Author و Title (از کلاس Book) بدون استفاده از نمونه ی کلاس Book، برای کلاس Exercise قابل دسترسی می باشد.

```
using System;
  using System;
  public class Book
  {
    public static string title;
    static public string author;
  }
  public class Exercise
  {
    static void Main()
    {
      Book.title = "Psychology and Human Evolution";
      Book.author = "Jeannot Lamm";
      Console.WriteLine("Book Characteristics");
      Console.WriteLine("Title: ");
      Console.WriteLine(Book.title);
```

```
        Console.WriteLine("Author: ");
        Console.WriteLine(Book.author);
        Book.title = "C# First Step";
        Book.author = "Alexandra Miles";
        Console.WriteLine("Book Characteristics");
        Console.WriteLine("Title: ");
        Console.WriteLine(Book.title);
        Console.WriteLine("Author: ");
        Console.WriteLine(Book.author);
        Console.ReadKey();
    }
}
```

به طور مشابه، می توان ترکیبی از متغیرهای ایستا و غیر ایستا برای کلاس تعریف کرد. فقط به خاطر داشته باشید که برای دسترسی به متغیر ایستا، شما ملزم به تعریف یا ایجاد نمونه ای از کلاس نیستید. در حالی که به منظور دسترسی به متغیر غیر ایستا، باید متغیری برای کلاس مورد نظر معرفی کنید.

متدهای ایستا

درست مشابه متغیر عضو، می توان متد کلاس را به عنوان ایستا معرفی کرد. متد مذکور قابلیت دسترسی به تمامی اعضای کلاس را دارد (البته سطح دسترسی متد به اعضای کلاس بستگی به این دارد که متغیر عضو چگونه تعریف شده باشد). به یاد داشته باشید که اعضای کلاس می توانند ایستا یا غیر ایستا باشند.

ایجاد متد ایستا

برای تعریف متدی به عنوان ایستا، کافی است کلیدواژه ی static را سمت چپ آن تایپ کنید.

مثال

```
public class Book
{
  static void CreateBook()
  {
  }
}
```

متد ایستا همچنین اجازه ی استفاده از تنظیم کننده ی سطح دسترسی (access modifier) را دارد. می توان کلیدواژه ی static را پیش یا پس از تنظیم کننده ی سطح دسترسی قرار دهید.

مثال

```
using System;
public class Book
{
  private static string title;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    static private string author;
    private static int pages;
    static private double price;
    static public void CreateBook()
    {
      title = "Psychology and Human Evolution";
      author = "Jeannot Lamm";
      pages = 472;
      price = 24.95;
    }
    internal static void ShowBook()
    {
      Console.WriteLine("Book Characteristics");
      Console.Write("Title: ");
      Console.WriteLine(Book.title);
      Console.Write("Author: ");
      Console.WriteLine(Book.author);
      Console.Write("Pages: ");
      Console.WriteLine(pages);
      Console.Write("Price: ");
      Console.WriteLine(price);
      Console.ReadKey();
    }
    public static int Main()
    {
      return 0;
    }
}
```

همان طور که برای فیلد ایستا (static field) ذکر شد، استفاده از متد ایستا نیز بستگی به مکان دسترسی آن دارد (منظور جایی است که متد ایستا در آن قابل دسترس است). به منظور دستیابی به عضو ایستا از طریق متد ایستای همان کلاس، می توان از اسم عضو ایستا استفاده کرد.

مثال

```
using System;
public class Book
{
  static string title;
  static string author;
  static int pages;
  static double price;
  static void CreateBook()
  {
    title = "Psychology and Human Evolution";
    author = "Jeannot Lamm";
    pages = 472;
    price = 24.95;
  }
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    static void ShowBook()
    {
      Console.WriteLine("Book Characteristics");
      Console.Write("Title: ");
      Console.WriteLine(Book.title);
      Console.Write("Author: ");
      Console.WriteLine(Book.author);
      Console.Write("Pages: ");
      Console.WriteLine(pages);
      Console.Write("Price: ");
      Console.WriteLine(price);
      Console.ReadKey();
    }
    public static int Main()
    {
      CreateBook();
      ShowBook();
      return 0;
    }
}
```

نتیجه ی زیر را به دست می دهد.

```
Book Characteristics
Title: Psychology and Human Evolution
Author: Jeannot Lamm
Pages: 472
Price: 24.95
```

همچنین می توان ابتدا اسم کلاس و به دنبال آن نقطه، عضو مورد نظر را را تایپ کرد. به مثال های زیر توجه کنید.

```
using System;
 public class Book
{
    . . . No Change
   static void CreateBook()
   {
      . . . No Change
   }
   static void ShowBook()
   {
      . . . No Change
   }
   public static int Main()
   {
     Book.CreateBook();
     Book.ShowBook();
     return 0;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    }
  }
```

به منظور دسترسی به عضوی ایستا از بیرون کلاس، ابتدا اسم کلاس، به دنبال آن نقطه و عضو موردنظر را تایپ کنید. نظر خود را به مثال های زیر جلب کنید.

```
using System;
  public class Book
  {
    . . . No Change
    static public void CreateBook()
    {
      . . . No Change
    }
    internal static void ShowBook()
    {
      . . . No Change
    }
  }
  public class Exercise
  {
    public static int Main()
    {
      Book.CreateBook();
      Book.ShowBook();
      return 0;
    }
  }
```

کلاس های ایستا

کلاس هم می تواند درست مثل متد و متغیر ایستا باشد. کلاس ایستا کلاسی هست که برای دسترسی به اعضای آن به هیچ وجه نیازی به نمونه ی کلاس نیست. به عبارتی روشن تر، دسترسی به اعضای کلاس ایستا فقط و فقط با استفاده از اسم کلاس (مستقیم از خود کلاس) و عملگر نقطه امکان پذیر می باشد.

کلاسی هست که اعضای آن باید ایستا باشند (در قالب ایستا ایجاد شوند). به عبارت دیگر، نمی توان عضوی غیر ایستا (non-static) به کلاسی ایستا افزود: کلیه ی اعضا به استثنای ثابت ها، باید ایستا باشند.

ایجاد کلاس ایستا

برای ایجاد کلاسی ایستا، کلیدواژه ی static را پیش از کلیدواژه ی class به کار ببرید.

مثال

```
using System;
  public static class Square
```

```
{
  public static double side;
  public static double Perimeter()
  {
    return side * 4;
  }
  public static double Area()
  {
    return side * side;
  }
}
public class Exercise
{
  public static int Main()
  {
    Square.Side = 36.84;
    Console.WriteLine("Square Characteristics");
    Console.Write("Side:    ");
    Console.WriteLine(Square.side);
    Console.Write("Perimeter: ");
    Console.WriteLine(Square.Perimeter());
    Console.Write("Area:    ");
    Console.WriteLine(Square.Area());
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه ی زیر را به دست می دهد.

```
Square Characteristics
Side:    36.84
Perimeter: 147.36
Area:    1357.1856
Press any key to continue . . .
```

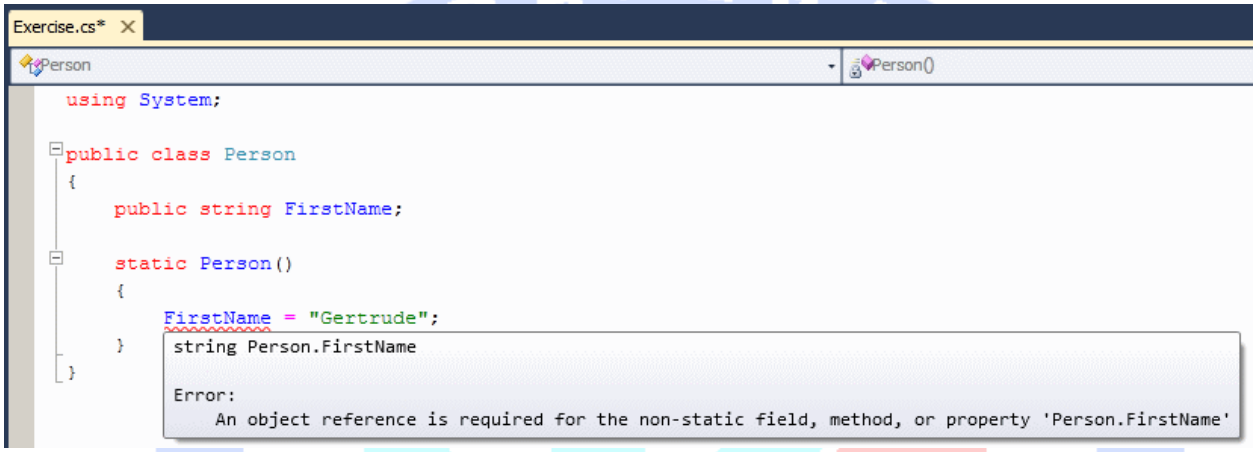
توابع سازنده ی Static

می توان constructor را مانند دیگر توابع معمول در C#، Static تعریف کرد. برای این منظور قوانینی وجود دارد که می بایست از آن ها پیروی کنید. برای استفاده از تابع سازنده ای که static تعریف شده (برای استفاده از آن خارج از کلاس خودش نیازی به ساختن نمونه یا شی از آن کلاس نیست)، بایستی آن را به طور صریح ایجاد نمایید (کامپایلر خودکار آن را برای شما ایجاد نمی کند).

تابع سازنده ی static باید به عنوان constructor پیش فرض مورد استفاده قرار گیرد، به عبارتی روشن تر باید constructor ای تعریف کنید که آرگومان نمی گیرد. مثال:

```
public class Person
{
  public string firstName;
  static Person()
  {
  }
}
```

اگر یک تابع سازنده ی static ایجاد کنید، آن وقت دیگر امکان استفاده از فیلدها یا اعضای غیر static کلاس مورد نظر وجود ندارد:



با این حال، شما این اجازه را دارید که به فیلدهای کلاس دسترسی داشته باشید. مثال:

```
using System;
  public class Person
  {
    public string firstName;
    static Person()
    {
    }
  }
  public class Exercise
  {
    public static int Main()
    {
      Person pers = new Person();
      pers.firstName = "Gertrude";
      Console.WriteLine("Personal Identification");
      Console.Write("Name: ");
      Console.WriteLine(pers.firstName);
      Console.ReadKey();
      return 0;
    }
  }
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

برای استفاده از constructor ای که دارای خاصیت static می باشد، چند گزینه پیشرو دارید. برای مقداردهی اولیه ی متغیر عضوی در تابع سازنده ی static، ابتدا لازم است یک متغیر برای آن کلاس اعلان کرده، سپس به آن متغیر عضو دسترسی پیدا کنید. مثال:

```
public class Person
{
  public string firstName;
  static Person()
  {
    Person pers = new Person();
    pers.firstName = "Gertrude";
  }
}
```

در واقع، یکی از دلایل استفاده از تابع سازنده ی static، مقداردهی اولیه ی فیلدهای static کلاس و یا انجام عملیاتی است که بین تمامی نمونه های کلاس به اشتراک گذاشته می شود. بنابراین، دیگر مورد استفاده ی تابع سازنده، مقداردهی اولیه ی متغیرهای عضو static آن کلاس می باشد. حال، در زمان دسترسی به فیلدهای static که در تابع سازنده مقداردهی اولیه شدند، می بینید که آن فیلدها مقادیر تخصیص یافته را درخود ذخیره کرده و آماده ی استفاده دارند.

```
using System;
public class Person
{
  public static string firstName;
  static Person()
  {
    firstName = "Gertrude";
  }
}
public class Exercise
{
  public static int Main()
  {
    Console.WriteLine("Personal Identification");
    Console.Write("Name: ");
    Console.WriteLine(Person.firstName);
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه ی زیر را به دنبال دارد:

```
Personal Identification
Name: Gertrude
Press any key to continue . . .
```

از آنجایی که تابع سازنده static می باشد، دسترسی به آن از طریق اعلان متغیر برای کلاس مورد نظر امکان پذیر نمی باشد.

قانون دیگری که در کار با تابع سازنده ی static باید رعایت کرد این است که نباید تحت هیچ شرایطی به آن تنظیم کننده ی سطح دسترسی (access modifier) اضافه کرد. کد زیر با خطا مواجه می شود:

```
public class Person
{
  public static Person()
  {
  }
}
```

می توانید یک کلاس تعریف کنید که ترکیبی از تابع سازنده ی static و یک یا چند تابع constructor غیر static می باشد. مثال:

```
public class Person
{
  public string firstName;
  public string lastName;
  static Person()
  {
  }
  public Person(string first, string last)
  {
    firstName = first;
    lastName = last;
  }
}
```

اگر چنین کلاسی تعریف کرده و متغیری برای آن اعلان کنید، constructor پیش فرضی برای آن وجود نداشته یا در حال حاضر دسترس پذیر نمی باشد. اگر هم می خواهید متغیری برای آن تعریف کنید، بایستی از یک constructor که دو پارامتر ورودی می گیرد (آن متغیر را به عنوان آرگومان می پذیرد)، استفاده نمایید. مثال:

```
using System;
public class Person
{
  public string firstName;
  public string lastName;
  static Person()
  {
  }
  public Person(string first, string last)
  {
    firstName = first;
    lastName = last;
  }
}
public class Exercise
{
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    public static int Main()
    {
      Person pers = new Person("Gertrude", "Monay");
      Console.WriteLine("Personal Identification");
      Console.Write("First Name: ");
      Console.WriteLine(pers.firstName);
      Console.Write("Last Name:  ");
      Console.WriteLine(pers.lastName);
      Console.ReadKey();
      return 0;
    }
  }
```

خروجی:

```
Personal Identification
First Name: Gertrude
Last Name:  Monay
Press any key to continue . . .
```

با توجه به آنچه گفته شد، تنها زمانی مجاز به تعریف constructor هستید که دلیل خوبی برای ایجاد آن داشته باشید.

محدوده و طول عمر یک متغیر

نمونه های زیادی از تعریف متغیرهای محلی را مشاهده کردیم.

مثال

```
using System;
public class Exercise
{
  public static int Main()
  {
    int number;
    return 0;
  }
}
```

در برخی موارد، ممکن است متغیری نیاز داشته باشید که توسط متدهای مختلف یک کلاس قابل دسترسی و اصلاح باشند. برای این منظور، می توانید متغیر مورد نظر را بیرون از محدوده ی (تمامی) متدها تعریف کنید. به چنین متغیری، متغیر سراسری (global variable) می گویند.

ایجاد و استفاده از متغیر سراسری

همان طور که پیش تر ذکر شد، برای ایجاد متغیر سراسری، باید آن را (در) بیرون متدها ولی در محدوده ی کلاس تعریف کرد.

مثال

```
using System;
public class Exercise
{
  int number;
  public static int Main()
  {
    return 0;
  }
}
```

پس از تعریف متغیر، می توان از هر متدی داخل کلاس برای دسترسی به آن استفاده کرد، از قبیل اصلاح مقدار آن. مثال های زیر را در نظر بگیرید.

```
using System;
public class Exercise
{
  int number;
  public void Modify()
  {
    number = 28;
  }
  public void Change()
  {
    number = 405;
  }
  public static int Main()
  {
    Exercise exo = new Exercise();
    Console.Write("Number: ");
    Console.WriteLine(exo.number);
    exo.Modify();
    Console.Write("Number: ");
    Console.WriteLine(exo.number);
    exo.Change();
    Console.Write("Number: ");
    Console.WriteLine(exo.number);
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه ی زیر به دست می آید.

```
Number: 0
Number: 28
Number: 405
Press any key to continue...
```

همچنین می توان از متد برای نشان دادن مقدار متغیر سراسری بهره گرفت.

```
using System;
public class Exercise
{
  int number;
  public void Modify()
  {
    number = 28;
  }
  public void Change()
  {
    number = 405;
  }
  public void Show()
  {
    Console.Write("Number: ");
    Console.WriteLine(number);
    Console.ReadKey();
  }
  public static int Main()
  {
    Exercise exo = new Exercise();
    exo.Show();
    exo.Modify();
    exo.Show();
    exo.Change();
    exo.Show();
    return 0;
  }
}
```

متغیرهای ایستا و سراسری

درست برخلاف C++، (در C#) امکان تعریف متغیر ایستا درون متد وجود ندارد. راه حل آن تعریف متغیر خارج از محدوده ی متد است، سپس می توان به متغیر مورد نظر از هر متدی که به آن نیاز دارد دست پیدا کرد.

مثال

```
using System;
public class Exercise
{
  static int number;
  public void Modify()
  {
    number = 28;
  }
  public void Change()
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    {
      number = 405;
    }
    public void Show()
    {
      Console.Write("Number: ");
      Console.WriteLine(number);
      Console.ReadKey();
    }
    public static int Main()
    {
      Exercise exo = new Exercise();
      exo.Show();
      exo.Modify();
      exo.Show();
      exo.Change();
      exo.Show();
      return 0;
    }
}
```

خصوصیات اعضای ایستا

ثابت ها

برخلاف زبان های C/C++، در C#، امکان ایجاد متغیر ثابت (constant variable) در کلاس وجود دارد. همان طور که در درس ۳ بحث کردیم برای تعریف متغیر ثابت، کافی است کلید واژه ی const را در سمت چپ متغیر مربوطه تایپ کنید. توجه داشته باشید که هنگام تعریف یک ثابت، باید آن را با مقدار مناسب و مقتضی مقداردهی اولیه کرد.

نمونه ی this

چنانچه کلاسی حاوی فیلد و متد باشد، عضوهای فیلد (غیر ایستا) خودکار در دسترس متدهای کلاس مورد نظر قرار می گیرند، حتی آن دسته فیلدهایی که خصوصی (private) هستند. هنگام دسترسی به فیلد یا متدی از متد دیگر (همان) کلاس، برای این که نشان دهیم عضوی که خواهان دسترسی به آن هستیم متعلق به همین کلاس است، از شی ای به نام this و به دنبال آن عملگر نقطه پیش از متد یا فیلد مورد نظر استفاده می کنیم (نمونه ی this در واقع ارجاع به کلاس جاری می باشد).

مثال

```
public class House
{
  internal char propertyType;
  internal uint bedrooms;
  internal void Display()
```

```
    {
        Console.WriteLine("=//= Altair Realtors =//=");
        Console.WriteLine("Properties Inventory");;
        Console.Write("Property Type: ");
        Console.WriteLine(this.propertyType);
        Console.Write("Bedrooms:     ");
        Console.WriteLine(this.bedrooms);
        Console.ReadKey();
    }
}
public class Exercise
{
    public static int Main()
    {
        House h = new House();
        h.Display();
        return 0;
    }
}
```

هنگام بکارگیری کلید واژه ی this پیروی از قوانین زیر الزامی می باشد.شی this هیچگاه تعریف نمی شود، بلکه هنگام ایجاد کلاس خودکار به آن اشاره می شود.

از this نمی توان در کلاس A استفاده کرد، سپس از آن برای دسترسی پیدا کردن به عضوی از کلاس B کمک گرفت.

به هیچ وجه نمی توان از شی this در متد static استفاده کرد.

بررسی اجمالی اعداد

زبان C#، برخلاف زبان های C، C++، Pascal، Visual Basic و Java، از ریاضی پشتیبانی نمی کند (امکان پشتیبانی توکار برای ریاضی ندارد) و مجبور است این قابلیت را از کتابخانه ها یا زبان های دیگر وام بگیرد.

برای انجام ساده ترین عملیات جبری و هندسی در C#، می توانید از متدهای کلاس Math.NET Framework استفاده کنید. همچنین می توان از کتابخانه ی بسیار غنی توابع visual basic برای این منظور کمک گرفت. کتابخانه ی بالا یکی از گسترده ترین مجموعه توابع حوضه های مختلف ریاضی تجاری را در بر دارد.

راه های مختلفی برای تعریف متغیر (از) نوع عددی وجود دارد که نمونه های آن را زیر مشاهده می کنید.

```
using System;
class Program
{
```

```
  static int Main()
  {
    short sNumber;
    int iNumber;
    double dNumber;
    decimal mNumber;
    return 0;
  }
}
```

علامت اعداد

یکی از قوانین پایه و اولیه ی C# این است که پس از تعریف متغیر و پیش از به کار بردن آن، متغیر باید مقداردهی اولیه شده باشد. نمونه های مقداردهی اولیه متغیر در مثال به نمایش گذاشته شده است.

```
using System;
class Program
{
  static int Main()
  {
    short sNumber = 225;
    int iNumber = -847779;
    double dNumber = 9710.275D;
    decimal mNumber = 35292742.884295M;
    Console.WriteLine("Short Integer:    {0}", sNumber);
    Console.WriteLine("Integral Number:  {0}", iNumber);
    Console.WriteLine("Double-Precision:  {0}", dNumber);
    Console.WriteLine("Extended Precision: {0}", mNumber);
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه ی زیر را به دست می دهد.

```
Short Integer:    225
Integral Number:  -847779
Double-Precision:  9710.275
Extended Precision: 35292742.884295
Press any key to continue...
```

هنگام مقداردهی اولیه متغیر با یک ثابت (constant)، در واقع مشخص می کنید که متغیر نام برده مثبت باشد یا منفی و یا ۰ که سرانجام علامت آن متغیر محسوب می شود. در صورتی که مقدار متغیر را از راه دیگری به دست آورید، ممکن است علامت آن مشخص نشود. اگر چه

می توان با استفاده از عملگر های مقایسه علامت متغیر (مثبت یا منفی بودن عدد) را به دست آورد، اما کلاس math متدی در اختیار شما قرار می دهد که این کار را برای شما انجام می دهد.

برای این منظور باید متد Math.Sign() را فراخواند. البته متد مذکور در نسخه های مختلف که دستور نحوی (syntax) آن ها به شکل زیر است اضافه بارگذاری شده.

```
public static int Sign(sbyte  value);
public static int Sign(short  value);
public static int Sign(int    value);
public static int Sign(long   value);
public static int Sign(sbyte  value);
public static int Sign(double value);
public static int Sign(decimal value);
```

هنگام فراخوانی این متد، متغیر یا مقدار دلخواه را به عنوان آرگومان ارسال کنید. متد نتایج زیر را بر می گرداند.

۱- چنانچه آرگومان منفی بود.

۰ چنانچه آرگومان ۰ بود.

۱ چنانچه آرگومان مثبت بود.

مثال های فراخوانی متد

```
using System;
class Program
{
  static int Main()
  {
    short sNumber = 225;
    int iNumber = -847779;
    double dNumber = 9710.275D;
    decimal mNumber = 35292742.884295M;
    Console.WriteLine("Number: {0} => Sign: {1}",sNumber, Math.Sign(sNumber));
    Console.WriteLine("Number: {0} => Sign: {1}",iNumber, Math.Sign(iNumber));
    Console.WriteLine("Number: {0} => Sign: {1}",dNumber, Math.Sign(dNumber));
    Console.WriteLine("Number: {0} => Sign: {1}\n",mNumber, Math.Sign(mNumber));
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه

```
Number: 225 => Sign: 1
umber: -847779 => Sign: -1
umber: 9710.275 => Sign: 1
umber: 35292742.884295 => Sign: 1
ress any key to continue...
```

بخش صحیح عدد ممیز شناور

همان طور که در درس ۳ تشریح کردیم، عدد ممیز شناور از دو بخش اصلی تشکیل شده، یک بخش عدد صحیح و یک بخش اعشار و این دو بخش مذکور توسط نقطه ی اعشار " . " از هم جدا می شوند. در برخی موارد (عملیات) لازم است بخش عدد صحیح یک مقدار را بدست آورید. کلاس Math در این زمینه کمک شایانی به شما می کند.

برای به دست آوردن بخش عدد صحیح یک رقم دهدهی، کلاس Math با متد Trancate() این امکان را برای شما فراهم می کند. متد بالا در دو نسخه ی زیر اضافه بار گذاری شده و دستور نحوی آن به شرح زیر است.

```
public static double Truncate(double d);
public static double Truncate(double d);
```

هنگام فراخوانی این متد، یک عدد یا متغیر از نوع شناور، دابل و دهدهی به آن ارسال کنید. متد نام برده بخش عدد صحیح یک مقدار را باز می گرداند.

مثال

```
using System;
class Program
{
  static int Main()
  {
    float number = 225.75f;
    Console.WriteLine("The integral part of {0} is {1}\n",number, Math.Truncate(number));
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه ی زیر به دست می آید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
The integral part of 225.75 is 225
Press any key to continue...
```

کمینه ی دو مقدار

می توان کمینه ی دو مقدار را بدون نیاز به نوشتن کد (خود) بدست آورد. برای این منظور کلاس Math مجهز به متدی است به نام .Min متد مزبور در ورژن های متفاوت اضافه بارگذاری شده که هر نسخه با نوع داده ی integral یا floating-point تطبیق داده شده است. دستورهای نحوی آن ها به شرح زیر است.

```
public static byte   Min(byte   val1, byte   val2);
public static sbyte  Min(sbyte  val1, sbyte  val2);
public static short  Min(short  val1, short  val2);
public static ushort Min(ushort val1, ushort val2);
public static int    Min(int    val1, int    val2);
public static uint   Min(uint   val1, uint   val2);
public static float  Min(float  val1, float  val2);
public static long   Min(long   val1, long   val2);
public static ulong  Min(ulong  val1, ulong  val2);
public static double Min(double val1, double val2);
public static decimal Min(decimal val1, decimal val2);
```

مثال

```
using System;
class Program
{
  static int Main()
  {
    int number1 = 8025;
    int number2 = 73;
    Console.WriteLine("The minimum of {0} and {1} is {2}", number1, number2, Math.Min(number1, number2));
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه ی زیر حاصل می گردد.

```
The minimum of 8025 and 73 is 73
Press any key to continue...
```

به خاطر داشته باشید که می توان از کلیدواژه های var و dynamic برای تعریف متغیرهای مورد نظر استفاده کرد :

```
using System;
class Program
{
  static int Main()
  {
    var number1 = 8025;
    dynamic number2 = 73;
    Console.WriteLine("The minimum of {0} and {1} is {2}",number1, number2, Math.Min(number1, number2));
    Console.ReadKey();
    return 0;
  }
 }
```

بیشینه ی مقدار integer یک سری

برای بدست آوردن بیشینه ی دو عدد، می توان متد Max() را (از کلاس Math) فراخواند. دستور نحوی این متد به ترتیب زیر است.

```
public static byte   Max(byte   val1, byte   val2);
public static sbyte  Max(sbyte  val1, sbyte  val2);
public static short  Max(short  val1, short  val2);
public static ushort Max(ushort val1, ushort val2);
public static int    Max(int    val1, int    val2);
public static uint   Max(uint   val1, uint   val2);
public static float  Max(float  val1, float  val2);
public static long   Max(long   val1, long   val2);
public static ulong  Max(ulong  val1, ulong  val2);
public static double Max(double val1, double val2);
public static decimal Max(decimal val1, decimal val2);
```

مثال فراخوانی متد بالا

```
using System;
class Program
{
  static int Main()
  {
    int number1 = 8025;
    int number2 = 73;
    Console.WriteLine("The maximum of {0} and {1} is {2}",number1, number2, Math.Max(number1, number2));
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه

```
The maximum of 8025 and 73 is 8025
Press any key to continue…
```

تبدیل مقادیر

تبدیل ضمنی

با نحوه ی تعریف نوع های رشته، ممیز شناور و صحیح (integral) آشنا شدیم. طریقه ی مقداردهی اولیه ی آن ها نیز تشریح شد. در صورت داشتن برنامه ای که متغیرهای مختلفی در آن به کار گرفته شده، می توان مقدار یک متغیر را به (مقدار) متغیر دیگری تبدیل کرد. در درس ا با مقدار حافظه ی مورد نیاز برای ذخیره سازی متغیر هر نوع داده آشنا شدیم.

Data Type	Name	Memory Size
byte	Byte	8 bits
sbyte	Signed Byte (بایت با علامت)	8 bits
char	Character	16 bits
short	Small Integer (عدد صحیح کوچک)	16 bits
ushort	Unsigned Small Integer	16 bits
int	Signed Integer(عدد صحیح علامت دار)	32 bits
uint	Unsigned Integer(عدد صحیح بدون علامت)	32 bits
float	Single-Precision Floating-Point Number (عدد با ممیز شناور با تنها یک رقم اعشار)	32 bits

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

double	Double-Precision Floating-Point Number (عدد با ممیز شناور با دو رقم اعشار)	64 bits
long	Signed Long Integer (عدد صحیح بزرگ علامت دار)	64 bits
ulong	Unsigned Long Integer	64 bits
decimal	Extended Precision Floating-Point Number(عدد ممیز اعشار با رقم اعشار طولانی)	128 bits

همان طور که در جدول بالا مشاهده می کنید، مقدار یک متغیر Byte می تواند در مقدار حافظه ی تخصیص یافته برای متغیر int جای گیرد و سرانجام مقدار متغیر int خود در متغیر long قابلیت ذخیره شدن را دارد. بر این اساس، می توان مقدار یک Byte را به متغیر int اختصاص داد و یا متغیر int را به متغیر long. همچنین به این خاطر که حافظه ی رزرو شده برای متغیر int بیشتر از مقدار حافظه ی تخصیص یافته به متغیر double می باشد، می توان متغیر اولی را به متغیر دومی اختصاص داد. مثال زیر را در نظر بگیرید.

```
using System;
class Program
{
  static int Main()
  {
    int iNumber = 2445;
    double dNumber = iNumber;
    Console.WriteLine("Number = {0}", iNumber);
    Console.WriteLine("Number = {0}\n", dNumber);
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه

```
Number = 2445
Number = 2445
Press any key to continue...
```

از این ویژگی با عنوان تبدیل ضمنی یاد می شود.

تبدیل صریح

به دلیل مقررات و الزامات حافظه، عکس فرایند تبدیل ضمنی امکان پذیر نمی باشد. به این خاطر که حافظه ی تخصیص یافته برای متغیر short کمتر از مقدار حافظه است که برای متغیر int رزرو شده، نمی توان مقدار متغیر int را به short اختصاص داد. برنامه ی زیر را در نظر بگیرید.

```
using System;
class Program
{
  static int Main()
  {
    int iNumber = 168;
    short sNumber = iNumber;
    Console.WriteLine("Number = {0}", iNumber);
    Console.WriteLine("Number = {0}\n", sNumber);
    Console.ReadKey();
    return 0;
  }
}
```

با اخطار زیر مواجه می شوید.

```
Cannot implicitly convert type 'int' to 'short'.
```

تبدیل (نوع) مقدار عبارتند از تبدیل یک مقدار به مقداری با نوع متفاوت. برای مثال، مقداری integer داریم و می خواهیم آن مقدار مورد نظر را در عبارتی که مقدار short می طلبد داشته باشیم. از این فرایند با عنوان تبدیل صریح نیز یاد می شود.

به منظور تبدیل یک مقدار یا متغیر، نوع داده ی دلخواه را پیش از مقدار مورد نظر داخل پرانتز تایپ می کنید.

مثال

```
using System;
class Program
{
  static int Main()
  {
    int iNumber = 168;
    short sNumber = (short)iNumber;
    Console.WriteLine("Number = {0}", iNumber);
```

```
    Console.WriteLine("Number = {0}\n", sNumber);
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه ی زیر حاصل می گردد.

```
Number = 168
Number = 168
Press any key to continue...
```

هنگام اجرای تبدیل صریح، توجه دقیق به نوع تبدیلی (نوعی که می خواهید تبدیل کنید) از اهمیت بسیار بالایی برخوردار است. اگر می خواهید مقدار integer به متغیر short تخصیص یابد، مقدار مورد نظر باید در 16 بیت جای گیرد، به این معنا که باید در محدوده (طیف) عددی 32768- تا ۳۲۷۶۷ قرار گیرد. هر مقدار دیگری فراتر از این رنج نتیجه ی غیرقابل پیش بینی بدست خواهد داد.

مثال

```
using System;
class Program
{
  static int Main()
  {
    int iNumber = 680044;
    short sNumber = (short)iNumber;
    Console.WriteLine("Number = {0}", iNumber);
    Console.WriteLine("Number = {0}\n", sNumber);
    Console.ReadKey();
    return 0;
  }
}
```

حاصل زیر به دست می آید.

```
Number = 680044
Number = 24684
Press any key to continue...
```

همان طور که مشاهده می کنید نتیجه ی نادرست حاصل شده.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

کلاس Convert

در درس ۱۳ و ۳۲ توضیح داده خواهد شد که هر نوع داده ی زبان C#، که خود از ساختار.NET Framework اقتباس شده، مجهز به متد ToString() است که قابلیت تبدیل مقدار نوع داده به نوع String را فراهم می کند. البته به امکان تبدیل مقداری از یک نوع اولیه (primitive type) به نوع اولیه ی دیگر پرداخته نشد. برای پشتیبانی از این امکان (تبدیل مقداری از یک نوع به نوع دیگر) .NET Framework کلاسی به نام Convert را ارائه می دهد. کلاس مذکور مجهز به چندین متد ایستا است که در این مبحث امکان تشریح همه ی آن ها وجود ندارد.

به خاطر داشته باشید که هر نوع داده ی اولیه ی C# از.NET Framework برگرفته شده است.

C# Data Type	Name	.NET Framework Structure
bool	Bollean	Boolean
byte	Byte	Byte
sbyte	Signed Byte	SByte
char	Character	Char
short	Small Integer	Int16
ushort	Unsigned Small Integer	UInt16
int	Integer	Int32
uint	Unsigned Integer	UInt32
long	Long Integer	Int64
ulong	Unsigned Long Integer	UInt64

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

float	Single-Precision Floating-Point	Single
double	Double-Precision Floating-Point	Double
decimal	Extended Precision Floating-Point Number	Decimal
No Explicit Type	Date/Time Value	DateTime
string	String	String

برای تطبیق دادن کلاس Convert به نوع داده های C#، کلاس مزبور مجهز به متد ایستایی است که نام آن با To آغاز شده و با اسم .NET Framework ساختارش پایان می یابد و به عنوان آرگومان، آن نوعی را می گیرد که باید تبدیل شود. بر این اساس، به منظور تبدیل عدد دهدهی از نوع double به عددی از نوع int، می توان متد ToInt32() را فراخواند سپس متغیر double را به عنوان آرگومان ارسال کرد. دستور نحوی آن به صورت زیر است.

```
public static int ToInt32(double value);
```

مثال

```
using System;
class Program
{
  static int Main()
  {
    double dNumber = 34987.68D;
    int iNumber = Convert.ToInt32(dNumber);
    Console.WriteLine("Number: {0}", dNumber);
    Console.WriteLine("Number: {0}", iNumber);
    Console.ReadKey();
    return 0;
  }
}
```

حساب (Arithmetic)

مقادیر مطلق

سیستم عدد اعشاری (decimal) از منفی بی نهایت آغاز شده و به مثبت بی نهایت ختم می شود. به عبارت دیگر، بسته به موقعیت آن ها (این که بعد از صفر قرار می گیرند یا قبل آن) اعداد یا منفی هستند یا مثبت. البته به جز عدد ۰ که خنثی تلقی می گردد. در برخی عملیات عدد حتماً باید مثبت باشد، حتی اگر در قالب منفی ارائه شده باشد. مقدار مطلق عدد x همان x است، البته در صورتی که عدد مورد نظر مثبت باشد. چنانچه عددی منفی بود، مقدار مطلق آن معادل مثبت آن عدد محسوب می شود. برای مثال، مقدار مطلق ۱۲ همان ۱۲ هست، این در حالی است که مقدار مطلق ۱۲- ، ۱۲ است.

برای به دست آوردن مقدار مطلق یک عدد، کلاس Math مجهز به متدی است به نام Abs، که در نسخه های متعدد اضافه بار گذاری (overload) شده است. دستور زبان (syntax) آن ها به شرح زیر است.

```
public static sbyte   Abs(sbyte   value);
public static short   Abs(short   value);
public static int     Abs(int     value);
public static float   Abs(float   value);
public static double  Abs(double  value);
public static long    Abs(long    value);
public static decimal Abs(decimal value);
```

متد فوق آرگومانی می گیرد که باید مقدار مطلق آن را به دست آورد :

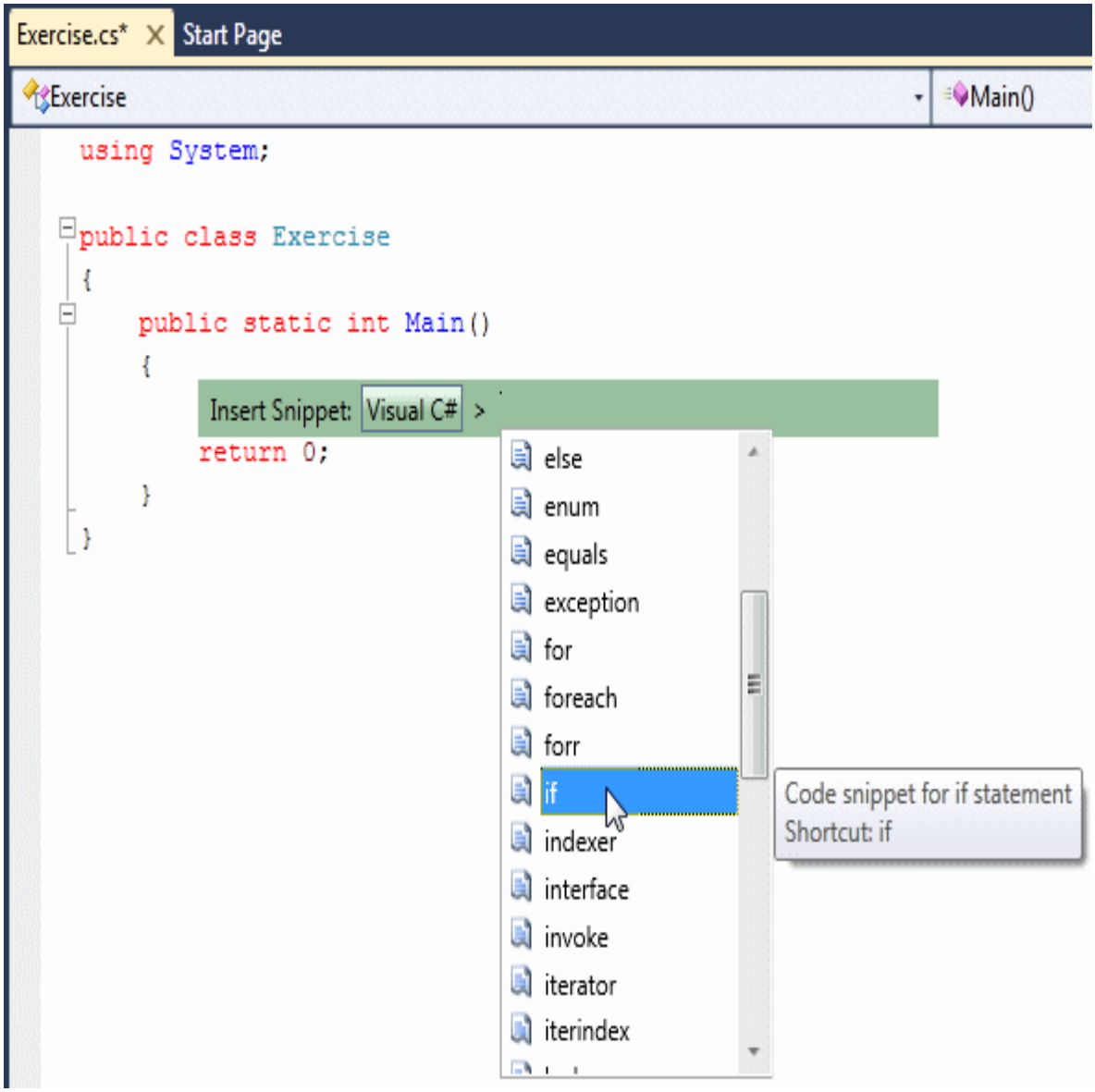
```
using System;
class Program
{
  static int Main()
  {
    int number = -6844;
    Console.WriteLine("Original Value = {0}", number);
    Console.WriteLine("Absolute Value = {0}\n", Math.Abs(number));
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه ی زیر را بدست می دهد.

```
Original Value = -6844
bsolute Value = 6844
ress any key to continue...
```

سقف (بیشترین مقدار) یک عدد

عددی با ممیز اعشار مثل ۱۲/۱۵۵ را در نظر بگیرید. این عدد بین integer 12 و integer 13 قرار دارد.



به همین نحو، عددی مثل -24.06 را در نظر بگیرید که منفی بوده و بین اعداد ۲۴- و ۲۵- قرار می گیرد. لازم به ذکر نیست که ۲۴- بزرگ تر است.

در علم حساب سقف یک عدد، نزدیک ترین عدد صحیح (integer) هست که از عدد مورد نظر بزرگ تر باشد. در مثال اول سقف ۱۲/۱۵۵، عدد ۱۳ است زیرا ۱۳ نزدیک ترین عدد صحیح بزرگتر از (یا برابر با) ۱۲/۱۵۵ می باشد. حال سقف عدد -24.06، ۲۴- می باشد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

برای به دست آوردن سقف یک عدد، کلاس Math مجهز به متدی است به نام Ceiling که در دو نسخه اضافه بارگذاری شده است. دستور نحوی این دو نسخه به صورت زیر می باشد.

```
public static double Ceiling(double a);
public static decimal Ceiling(decimal d);
```

متد بالا به عنوان آرگومان یک متغیر عدد با ممیز اعشار می گیرد که سقف آن را باید به دست آورد.

مثال

```
using System;
class Program
{
  static int Main()
  {
    double value1 = 155.55; double value2 = -24.06;
    Console.WriteLine("The ceiling of {0} is {1}",value1, Math.Ceiling(value1));
    Console.WriteLine("The ceiling of {0} is {1}\n",value2, Math.Ceiling(value2));
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه

```
The ceiling of 155.55 is 156
he ceiling of -24.06 is -24
ress any key to continue...
```

جدا از کلاس Math، ساختار Double پیاده سازی خودش از متد نام برده را با دستور نحوی زیر ارائه می دهد.

```
public static decimal Ceiling(decimal d);
```

کف (کمترین مقدار) یک عدد

دو عدد اعشاری 128.44 و 36.72- را در نظر بگیرید. 128.44 بین دو عدد 128 و 129 قرار دارد که ۱۲۸ عدد کوچکتر محسوب می شود. -36.72 بین دو عدد 37- و 36- قرار گرفته که 37- عدد کوچکتر محسوب می شود. کمترین یا پایین ترین ولی در عین حال نزدیک ترین مقدار integer یک عدد را کف آن عدد می گویند.

برای بدست آوردن کمترین مقدار یک عدد کلاس Math، متد Floor() را ارائه می دهد. این متد در دو نسخه ی زیر اضافه بارگذاری می شود.

```
public static double Floor(double d);
public static decimal Floor(decimal d);
```

متد Floor() مقدار مربوط را به عنوان آرگومان گرفته و عدد صحیحی (integer) را برمی گرداند که کوچکتر از (یا برابر با) آرگومان نام برده باشد.

مثال

```
using System;
class Program
{
  static int Main()
  {
    double value1 = 1540.25;
    double value2 = -360.04;
    Console.WriteLine("The floor of {0} is {1}",value1, Math.Floor(value1));
    Console.WriteLine("The floor of {0} is {1}\n",value2, Math.Floor(value2));
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه ی زیر حاصل می گردد.

```
The floor of 1540.25 is 1540
he floor of -360.04 is -361
ress any key to continue...
```

به جای استفاده از کلاس Math، می توان ساختار Double را به کار برد که متدی ویژه ی بدست آوردن کم ترین مقدار (کف) عدد اعشاری دارد. دستور نحوی آن به شکل زیر است :

```
public static decimal Ceiling(decimal d);
```

توان یک عدد

در C# توان با فرمول زیر انجام می شود :

$$ReturnValue = X^{y}$$

کلاس Math برای انجام عملیات توان، متد Pow را در اختیار شما قرار می دهد که دستور نحوی آن به شکل زیر است :

```
public static double Pow(double x, double y);
```

این متد دو آرگومان می گیرد. آرگومان اول، x، عدد پایه است که باید ارزیابی شود. آرگومان دوم، y، که توان نامیده می شود، آرگومان x را به توان آرگومان y می برد.

مثال

```
using System;
class Program
{
  static int Main()
  {
    const double source = 25.38;
    const double exp = 3.12;
    double result = Math.Pow(source, exp);
    Console.WriteLine("Pow({0}, {1}) = {2}\n",source, exp, result);
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه

```
Pow(25.38, 3.12) = 24099.8226934415
Press any key to continue...
```

مقدار نمایی (the Exponentional)

می توان مقدار نمایی یک عدد را محاسبه کرد. کلاس Math، متد Exp() را برای این منظور در نظر گرفته است. دستور نحوی آن به شکل زیر می باشد.

```
public static double Exp (double d);
```

مثال

```
using System;
class Program
{
  static int Main()
  {
    Console.WriteLine("The exponential of {0} is {1}",709.78222656, Math.Exp(709.78222656));
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه ی زیر به دست می آید :

```
The exponential of 709.78222656 is 1.79681906923757E+308
Press any key to continue...
```

چنانچه مقدار x از عدد (تقریبی) -708.395996093 کوچکتر بود، مقدار صفر برگردانده می شود و وضعیت پاریز (underflow) رخ می دهد. در صورتی که مقدار آرگومان x بزرگتر از عدد (تقریبی) 709.78222656 بود، (نتیجه) وضعیت سرریز (overflow) رخ می دهد.

لگاریتم طبیعی

برای محاسبه لگاریتم طبیعی یک عدد، می توان متد Math.Log()را فراخواند. متد مزبور به دو صورت زیر ارائه می شود. دستور نحوی یکی از آن دو

```
public static double Log(double d);
```

مثال

```
using System;
class Program
{
  static int Main()
  {
    double log = 12.48D;
    Console.WriteLine("Log of {0} is {1}", log, Math.Log(log));
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه ی زیر حاصل می شود.

```
Log of 12.48 is 2.52412736294128
Press any key to continue...
```

لگاریتم پایه ی ۱۰

متد Math.Log10() لگاریتم پایه ی ۱۰ یک عدد را محاسبه می کند. دستور نحوی متد نام برده به صورت زیر است.

```
public static double Log10(double d);
```

عددی که ارزیابی می شود به عنوان آرگومان ارسال می گردد. متد، لگاریتم مورد نظر را با فرمول زیر بر مبنای ۱۰ بر می گرداند.

$y = \log 10^{x}$

که معادل

$x = 10^{y}$

مثال

```
using System;
class Program
{
  static int Main()
  {
    double log10 = 12.48D;
    Console.WriteLine("Log of {0} is {1}", log10, Math.Log10(log10));
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه ی زیر حاصل می گردد.

```
Log of 12.48 is 1.09621458534641
Press any key to continue...
```

لگاریتم به هر مبنایی

متد Math.Log() نسخه ی دیگری فراهم می کند که دستور نحوی آن بدین صورت است.

```
public static double Log(double a, double newBase);
```

متغیری که مقدار لگاریتمی آن محاسبه می شود، به عنوان اولین آرگومان به متد ارسال می گردد. آرگومان دوم به شما اجازه می دهد مبنای (لگاریتم) دلخواه خود را تعیین کنید. چنین متدی از فرمول زیر استفاده می کند.

$Y = \log_{NewBase}{}^{x}$

که با فرمول زیر یکی است.

$x = NewBase^{y}$

مثال

```
using System;
class Program
{
  static int Main()
  {
    double logN = 12.48D;
    Console.WriteLine("Log of {0} is {1}", logN, Math.Log(logN, 4));
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه

```
Log of 12.48 is 1.82077301454376
Press any key to continue...
```

ریشه ی دوم / جذر

می توان ریشه ی مربع یک عدد مثبت اعشاری را به دست آورد. برای این منظور، کلاس Math مجهز به متدی است به نام Sqrt که دستور نحوی آن به شکل زیر می باشد.

```
public static double Sqrt(double d);
```

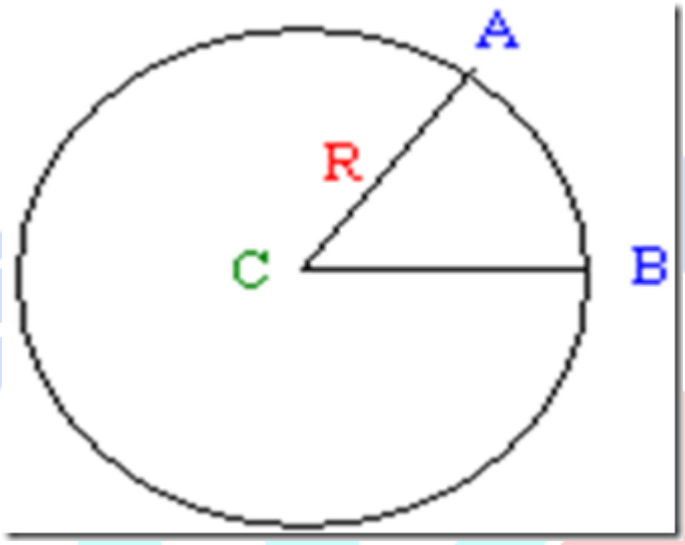
متد بالا یک آرگومان به عنوان عدد با ممیز شناور مثبت می گیرد. پس از محاسبه، متد ریشه ی مربع (جذر) x را برمی گرداند.

```
using System;
class Program
{
  static int Main()
  {
    double sqrt = 8025.73D;
    Console.WriteLine("The square root of {0} is {1}", sqrt, Math.Sqrt(sqrt));
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه

```
The square root of 8025.73 is 89.5864387058666
Press any key to continue...
```

مثلثات



دایره (مکان هندسی)، نقاطی از صفحه است که فاصله شان از نقطه ی ثابتی واقع در آن صفحه، مقدار ثابتی باشد. شعاع (r) خطی مستقیم است که مرکز دایره (C) را به نقطه ای از محیط دایره وصل می کند. خطی که نقاطی از صفحه که فاصله شان از نقطه ی ثابتی واقع در آن صفحه، مقدار ثابتی باشد را به هم وصل کند محیط دایره خوانده می شود. فاصله ی بین دو نقطه از محیط دایره تا مرکز آن قطر دایره نامیده می شود، به عبارت دیگر، قطر دایره دو برابر شعاع دایره می باشد.

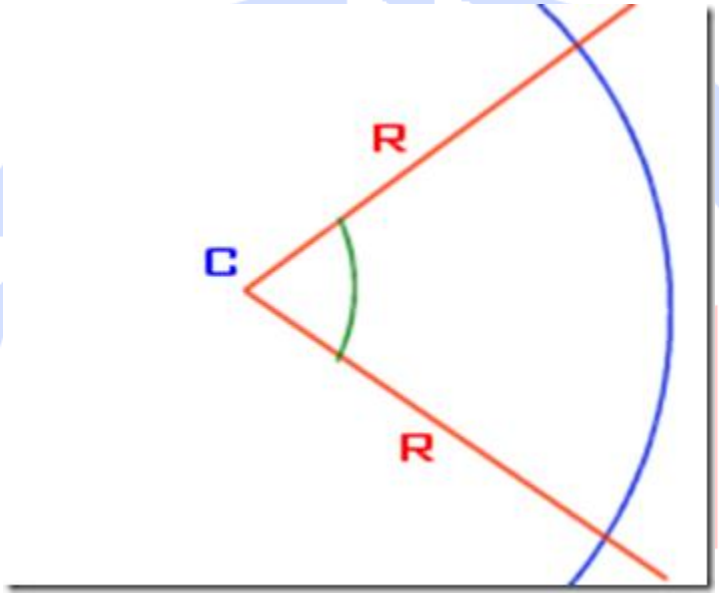
به منظور مدیریت اندازه گیری و دیگر عملیات مرتبط، محیط دایره به ۳۶۰ بخش تقسیم می شود. هر یک از بخش های بیان شده یک درجه اطلاق می گردد. واحد اندازه گیری آن درجه می باشد که با این علامت نشان داده می شود : " ° ". اندازه گیری دو نقطه ی A و D محیط دایره شامل ۱۵ بخش می شود. در این مثال، اندازه گیری مورد نظر به صورت 15° نمایش داده می شود.

فاصله ی بین دو نقطه ی A و B شکلی گرد است که در هندسه کمان (arc) خوانده می شود. زاویه نسبت فصله ی بین دو نقطه A و B محیط دایره است که توسط شعاع دایره (r) تقسیم شده (به عبارت دیگر از برخورد دو نیم خط زاویه تشکیل می شود). تعریف فوق را می توان به صورت فرمول زیر درآورد :

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

$$\theta = \frac{AB}{R}$$

بنابراین، زاویه نسبت کمان بر شعاع دایره می باشد. به این خاطر که زاویه یک نسبت است نه اندازه گیری هندسی (به عبارت روشن تر زاویه یک بعد نیست)، کاملاً مستقل از اندازه ی یک دایره محسوب می شود. بدیهی است که این زاویه نشانگر تعداد بخش هایی می باشد که توسط سه نقطه ی اشغال شده. واحد دیگر (یا شاید بگیم بهتر) برای اندازه گیری و سنجش زاویه رادیان (radian) می باشد.



دَوَران (cycle) واحد اندازه گیری چرخش، گِردِ یک دایره است. به این خاطر که چرخش لزوماً کامل صورت نمی گیرد، بسته به شرایط، سنجش بر اساس زاویه ای انجام می شود که در گردش طی شده. برای مثال ممکن است دَوَران تنها بخشی از یک دایره را طی کند که در آن صورت چرخش کامل نخواهد بود. همچنین ممکن است دوران، کل مساحت یک دایره را طی کند (360°). دَوَران برابر است با رادیان تقسیم بر ۲.

ثابت PI

کلمه ی π، که به صورت Pi نیز نوشته می شود، عدد ثابتی است که در محسابات گوناگون در علم ریاضی به کار می رود. مقدار تقریبی آن معادل 3.1415926535897932 می باشد. ماشین حساب سیستم عامل ویندوز آن را بدین صورت نمایش می دهد :

3.14159265358979323846264338327950

برای پشتیبانی از قابلیت عدد Pi، کلاس Math مجهز به ثابتی به نام PI می باشد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

یک قطر در واقع دوبرابر شعاع دایره است. در علم هندسه، بدین صورت نوشته می شود : 2R. در زبان برنامه نویسی C++ قطر به صورت 2 * R یا R * 2 نمایش داده می شود (به این خاطر که ضرب متقارن و برابر است). محیط یک دایره از ضرب (کردن) قطر آن در ثابت Pi به دست می آید که به صورت های زیر نوشته می شود.

2Rπ، یا 2 * R * π و یا 2 * R * Pi

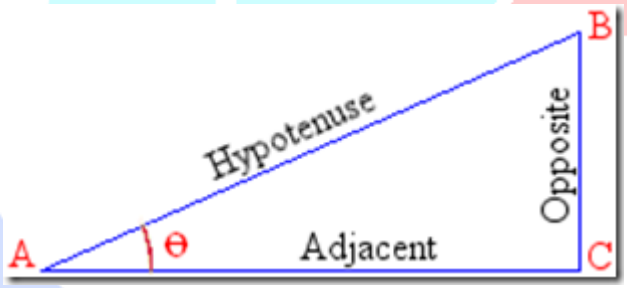
یک رادیان از فرمول 2Rπ/R یا 2Rπ/R rad به دست می آید که برابر است با 2π یا 2 * Pi rad.

برای انجام عملیات تبدیل بین درجه و رادیان (تبدیل درجه به رادیان)، می توان از فرمول زیر استفاده کرد :

360° = 2π rad که برابر است با 1 rad = 360° / 2π = 57.3°.

کسینوس یک مقدار

شکل هندسی زیر را در نظر بگیرید.



حال AB را، طول نقطه ی A تا B در نظر بگیرید که از آن به عنوان وتر (hypotenuse) یاد می شود. همچنین تصور کنید AC طول نقطه ی A تا C است که ضلع مجاور (Adjacent) نقطه ی A نیز خوانده می شود. کسینوس زاویه ی نقطه ی A نسبت AC/AB می باشد. به عبارت دیگر، نسبت ضلع مجاور، AC، بر وَترAB

مقدار باز گردانده شده حاصل تقسیم، عددی با دو رقم اعشار بین 1- و ۱ می باشد.

برای محاسبه ی کسینوس یک زاویه، کلاس Math متد Cos() را به کار می برد. دستور نحوی آن به شکل زیر است.

```
public static double Cos(double d);
```

مثال

```
using System;
class Program
{
  static int Main()
  {
    int number = 82;
    Console.WriteLine("The cosine of {0} is {1}", number, Math.Cos(number));
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه

```
The cosine of 82 is 0.949677697882543
Press any key to continue...
```

سینوس یک مقدار

AB را طول نقطه ی A تا B در نظر بگیرید که همان وتر نام دارد. همچنین CB را در نظر بگیرید، طول نقطه ی C تا نقطه ی B، که ضلع مقابل نامیده می شود. سینوس یک مقدار عبارتند از نسبت CB/AB؛ به عبارت دیگر، نسبت ضلع مقابل، CB بر وتر AB.

برای محاسبه ی سینوس یک مقدار، می توان متد Sin() را از کلاس Math فراخواند. دستور نحوی آن

```
public static double Sin(double a);
```

مثال

```
using System;
class Program
{
  static int Main()
  {
    double number = 82.55;
    Console.WriteLine("The sine of {0} is {1}", number, Math.Sin(number));
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه

```
The sine of 82.55 is 0.763419622322519
Press any key to continue...
```

تانژانت

AC را طول نقطه ی A تا C و BC را طول نقطه ی B تا C در نظر بگیرید. تانژانت نتیجه ی BC/AC هست؛ به عبارت روشن تر، نسبت BC بر AC. برای کمک در محاسبه ی تانژانت یک عدد، کلاس Math مجهز به متدی است به نام Tan که دستور نحوی آن به شکل زیر می باشد.

```
public static double Tan(double a);
```

مثال

```
using System;
class Program
{
  static int Main()
  {
    uint number = 225;
    Console.WriteLine("The tangent of {0} is {1}", number, Math.Tan(number));
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه

```
The tangent of 225 is -2.53211499233434
Press any key to continue...
```

Arc Tangent

BC را طول نقطه ی B تا C و AC را طول نقطه ی A تا نقطه ی B در نظر بگیرید. arc tangent نسبت BC/AC است. به منظور محاسبه ی arc tangent یک مقدار، می توان متد Math.Atan() را مورد استفاده قرار داد. دستور نحوی

```
public static double Atan(double d);
```

مثال

```
using System;
class Program
{
  static int Main()
  {
    short number = 225;
    Console.WriteLine("The arc tangent of {0} is {1}",number, Math.Atan(number));
    Console.ReadKey();
    return 0;
  }
```

```
}
```

نتیجه

```
The arc tangent of 225 is 1.56635191161394
Press any key to continue…
```

خواندن وقالب بندی داده ها

درخواست (دریافت) مقدار رشته

در اغلب انتساب های (که به آن دستور assignment/ تساوی نیز می گویند و با علامت = نشان داده می شود و کار آن کپی مقدار در متغیر است) کدنویسی، هنگام نوشتن برنامه ی کاربردی از مقدار رشته (string value) خبر نداریم. به منظور درخواست دریافت مقدار رشته (یا هر متغیر دیگری که در این مبحث به آن می پردازیم)، باید توابع Console.Read() یا Console.ReadLine() را فراخوانده، سپس تابع مذکور را به اسم متغیری که می خواهید مقدار آن را بازیابی کنید، اختصاص دهید.

مثال

۱. برنامه ی Microsoft Visual Studio را اجرا کنید.
۲. برای راه اندازی پروژه ی جدید به فهرست گزینه ی اصلی برنامه مراجعه کرده، گزینه ی File -> New Project را انتخاب کنید.
۳. در لیست میانی روی Empty Project کلیک کنید.
۴. اسم پروژه را به gdcs5 تغییر دهید.
۵. حال، ok را کلیک کنید.
۶. در پنجره ی Solution Explorer، روی gdcs5 راست کلیک کرده، سپس : gdcs5 -> Add -> New Item…
۷. گزینه ی Code File را از لیست میانی آن انتخاب کنید.
۸. اسم مورد نظر را به ClearningOrder تغییر داده و کلید Enter را بزنید.
۹. اکنون، به منظور درخواست رشته ها از کاربر، فایل مورد نظر را به صورت زیر اصلاح کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
using System;
public class CleaningOrder
{
  public static int Main()
  {
    string customerName, homePhone;
    Console.Title = "Georgetown Dry Cleaning Services";
    Console.WriteLine("-/- Georgetown Dry Cleaning Services -/-");
    // Request customer information from the user
    Console.Write("Enter Customer Name: ");
    customerName = Console.ReadLine();
    Console.Write("Enter Customer Phone: ");
    homePhone = Console.ReadLine();
    Console.Clear();
    // Display the receipt
    Console.WriteLine("=====================================");
    Console.WriteLine("-/- Georgetown Dry Cleaning Services -/-");
    Console.WriteLine("=====================================");
    Console.Write("Customer:  ");
    Console.WriteLine(customerName);
    Console.Write("Home Phone: ");
    Console.WriteLine(homePhone);
    Console.WriteLine("=====================================\n");
    System.Console.ReadKey();
    return 0;
  }
}
```

۱۰. برنامه را اجرا کنید. هنگامی که از شما نام مشتری درخواست می شود، اسم مورد نظر را وارد کنید مثلاً James Watson

۱۱. کلید Enter را بزنید.

۱۲. حال، شماره ی تلفن همراه مورد نظر را وارد می کنیم مثلاً (410) 493-2005 .

```
-/- Georgetown Dry Cleaning Services -/-
Enter Customer Name: James Watson
Enter Customer Phone: (410) 493-2005
```

۱۳. کلید Enter را فشار دهید.

```
=====================================
/- Georgetown Dry Cleaning Services -/-
=====================================
ustomer:  James Watson
ome Phone: (410) 493-2005
=====================================
```

۱۴. اکنون پنجره ی DOS را بسته و به محیط برنامه نویسی بازگردید.

درخواست عدد

در زبان C# هر آنچه تایپ می کنید، یک رشته محسوب می شود و تا زمانی که شما از compiler درخواست نکنید، رشته های مزبور تحلیل و پردازش نمی شوند. بنابراین، برای دریافت عدد مورد نظر از کاربر، ابتدا باید رشته را درخواست کنید.

مثال

```
using System;
public class Exercise
{
  public static void Main()
  {
    int number;
    string strNumber;
    strNumber = Console.ReadLine();
  }
}
```

پس از دریافت رشته، باید آن را به عدد تبدیل کرد. برای این منظور هر یک از نوع داده های.NET Framework مکانیسمی به نام Parse را دارند. برای به کار بردن تابع Parse()، نوع داده و به دنبال آن عملگر نقطه "." و پرانتز را تایپ کنید. در پرانتز های parse، رشته ای را که از کاربر درخواست کرده بودید، وارد کنید.

مثال

```
using System;
public class Exercise
{
  public static void Main()
  {
    int number;
    string strNumber;
    strNumber = Console.ReadLine();
    number = int.Parse(strNumber);
  }
}
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

روشی پیچیده ولی سریع تر، تایپ کردن دستور Console.ReadLine() در پرانتزهای تابع Parse است که همان نتیجه را به دست می دهد.

مثال

```
using System;
public class Exercise
{
  public static int Main()
  {
    int number;
    number = int.Parse(Console.ReadLine());
    return 0;
  }
}
```

خواندن مقادیر عددی

۱. برای دریافت اعداد مورد نظر از کاربر، فایل را به صورت زیر تغییر دهید.

```
using System;
  public class CleaningOrder
  {
    public static int Main()
    {
      // Price of items
      const double PriceOneShirt = 0.95;
      const double PriceAPairOfPants = 2.95;
      const double PriceOneDress = 4.55;
      const double TaxRate = 0.0575;  // 5.75%
      // Customer personal infoirmation
      string customerName, homePhone;
      // Unsigned numbers to represent cleaning items
      uint numberOfShirts, numberOfPants, numberOfDresses;
      // Each of these sub totals will be used for cleaning items
      double subTotalShirts, subTotalPants, subTotalDresses;
      // Values used to process an order
      double totalOrder, taxAmount, salesTotal;
      double amountTended, moneyChange;
      Console.Title = "Georgetown Dry Cleaning Services";
      Console.WriteLine("-/- Georgetown Dry Cleaning Services -/-");
      // Request customer information from the user
      Console.Write("Enter Customer Name: ");
      customerName = Console.ReadLine();
      Console.Write("Enter Customer Phone: ");
      homePhone = Console.ReadLine();
```

```
// Request the quantity of each category of items
Console.Write("Number of Shirts: ");
string strShirts = Console.ReadLine();
numberOfShirts = uint.Parse(strShirts);
Console.Write("Number of Pants: ");
string strPants = Console.ReadLine();
numberOfPants = uint.Parse(strPants);
Console.Write("Number of Dresses: ");
string strDresses = Console.ReadLine();
numberOfDresses = uint.Parse(strDresses);
// Perform the necessary calculations
subTotalShirts = numberOfShirts * PriceOneShirt;
subTotalPants = numberOfPants * PriceAPairOfPants;
subTotalDresses = numberOfDresses * PriceOneDress;
// Calculate the "temporary" total of the order
totalOrder = subTotalShirts + subTotalPants +subTotalDresses;
// Calculate the tax amount using a constant rate
taxAmount = totalOrder * TaxRate;
// Add the tax amount to the total order
salesTotal = totalOrder + taxAmount;
// Communicate the total to the user...
Console.Write("\nThe Total order is: ");
Console.WriteLine(salesTotal);
// and request money for the order
Console.Write("Amount Tended? ");
amountTended = double.Parse(Console.ReadLine());
// Calculate the difference owed to the customer
// or that the customer still owes to the store
moneyChange = amountTended - salesTotal;
Console.Clear();
// Display the receipt
Console.WriteLine("=====================================");
Console.WriteLine("-/- Georgetown Dry Cleaning Services -/-");
Console.WriteLine("=====================================");
Console.Write("Customer: ");
Console.WriteLine(customerName);
Console.Write("Home Phone: ");
Console.WriteLine(homePhone);
Console.WriteLine("-----------------------------------");
Console.WriteLine("Item Type  Qty Unit/Price Sub-Total");
Console.WriteLine("-----------------------------------");
Console.Write("Shirts    ");
Console.Write(numberOfShirts);
Console.Write("    ");
Console.Write(PriceOneShirt);
Console.Write("    ");
Console.WriteLine(subTotalShirts);
Console.Write("Pants     ");
Console.Write(numberOfPants);
Console.Write("    ");
```

```
        Console.Write(PriceAPairOfPants);
        Console.Write("    ");
        Console.WriteLine(subTotalPants);
        Console.Write("Dresses    ");
        Console.Write(numberOfDresses);
        Console.Write("    ");
        Console.Write(PriceOneDress);
        Console.Write("    ");
        Console.WriteLine(subTotalDresses);
        Console.WriteLine("-----------------------------------");
        Console.Write("Total Order:    ");
        Console.WriteLine(totalOrder);
        Console.Write("Tax Rate:      ");
        Console.Write(TaxRate * 100);
        Console.WriteLine('%');
        Console.Write("Tax Amount:     ");
        Console.WriteLine(taxAmount);
        Console.Write("Net Price:      ");
        Console.WriteLine(salesTotal);
        Console.WriteLine("-----------------------------------");
        Console.Write("Amount Tended:  ");
        Console.WriteLine(amountTended);
        Console.Write("Difference:     ");
        Console.WriteLine(moneyChange);
        Console.WriteLine("=====================================");
        System.Console.ReadKey();
        return 0;
    }
}
```

۲. برنامه را اجرا کنید.

مثال

```
-/- Georgetown Dry Cleaning Services -/-
    Enter Customer Name: Genevieve Alton
    Enter Customer Phone: (202) 974-8244
    Number of Shirts: 8
    Number of Pants:  2
    Number of Dresses: 3
    The Total order is: 28.711125
    Amount Tended? 30
```

۳. حال، کلید Enter را بزنید.

```
=================================
    -/- Georgetown Dry Cleaning Services -/-
    =====================================
```

```
Customer: Genevieve Alton
Home Phone: (202) 974-8244
-------------------------------------
Item Type  Qty Unit/Price Sub-Total
-------------------------------------
Shirts    8    0.95    7.60
Pants     2    2.95    5.90
Dresses   3    4.55    13.65
-------------------------------------
Total Order:    27.15
Tax Rate:       5.7500%
Tax Amount:     1.561125
Net Price:      28.711125
-------------------------------------
Amount Tended: 30
Difference:    1.288875
=====================================
```

۴. پنجره ی DOS را ببندید.

درخواست تاریخ و زمان

همان فرایندی که برای دریافت هر رقم دیگری انجام می شود، می توان برای درخواست مقدار تاریخ پیاده کرد. برای این منظور ابتدا لازم است رشته را از کاربر درخواست کنید.

```
using System;
 namespace ValueRequests
 {
   class Exercise
   {
     static void Main()
     {
       string strDateHired;
       strDateHired = Console.ReadLine();
     }
   }
 }
```

پس از اینکه کاربر رشته ی مربوطه را وارد کرد، می توانید آن را به مقدار DateTime تبدیل کنید. درست مثل هر مقدار دیگری که از کاربر درخواست می کنید، مقدار تاریخ یا زمان وارد شده باید معتبر و صحیح باشد، در غیر این صورت برنامه پیغام خطا می دهد. به این خاطر که مقادیر تاریخ و زمان قوانین خاصی برای قالب بندی (format) دارند، باید سعی کنید به کاربر بفهمانید که مقادیر نام برده را با فرمت لازم و مناسب وارد کنند.

به صورت پیش فرض، چنانچه از کاربر تنها مقدار تاریخی (معتبر) را تقاضا کنید، (پس از اینکه کاربر تاریخ مورد نظر را تایپ کرد) compiler به آن مقدار midnight را اضافه می کند. در صورتی که از کاربر فقط مقدار زمانی (معتبر) را درخواست کنید، Compiler تاریخ جاری را به مقدار فوق اضافه می کند.

درخواست مقادیر تاریخ و زمان

۱. به منظور درج تاریخ و مقدار جدید، برنامه را به صورت زیر اصلاح کنید.

```
using System;
  public class OrderProcessing
  {
    public static int Main()
    {
      // Price of items
      const double PriceOneShirt = 0.95;
      const double PriceAPairOfPants = 2.95;
      const double PriceOneDress = 4.55;
      const double TaxRate = 0.0575;  // 5.75%
      // Basic information about an order
      string customerName, homePhone;
      DateTime orderDate;
      // Unsigned numbers to represent cleaning items
      uint numberOfShirts, numberOfPants, numberOfDresses;
      // Each of these sub totals will be used for cleaning items
      double subTotalShirts, subTotalPants, subTotalDresses;
      // Values used to process an order
      double totalOrder, taxAmount, salesTotal;
      double amountTended, moneyChange;
      Console.Title = "Georgetown Dry Cleaning Services";
      Console.WriteLine("-/- Georgetown Dry Cleaning Services -/-");
      // Request order information from the user
      Console.Write("Enter Customer Name: ");
      customerName = Console.ReadLine();
      Console.Write("Enter Customer Phone: ");
      homePhone = Console.ReadLine();
      Console.WriteLine("Enter the order date and " + "time (mm/dd/yyyy hh:mm AM/PM)");
      orderDate = DateTime.Parse(Console.ReadLine());
      // Request the quantity of each category of items
      Console.Write("Number of Shirts: ");
      string strShirts = Console.ReadLine();
      numberOfShirts = uint.Parse(strShirts);
      Console.Write("Number of Pants:  ");
      string strPants = Console.ReadLine();
      numberOfPants = uint.Parse(strPants);
      Console.Write("Number of Dresses: ");
      string strDresses = Console.ReadLine();
```

```
numberOfDresses = uint.Parse(strDresses);
// Perform the necessary calculations
subTotalShirts = numberOfShirts * PriceOneShirt;
subTotalPants = numberOfPants * PriceAPairOfPants;
subTotalDresses = numberOfDresses * PriceOneDress;
// Calculate the "temporary" total of the order
totalOrder = subTotalShirts +subTotalPants + subTotalDresses;
// Calculate the tax amount using a constant rate
taxAmount = totalOrder * TaxRate;
// Add the tax amount to the total order
salesTotal = totalOrder + taxAmount;
// Communicate the total to the user…
Console.Write("\nThe Total order is: ");
Console.WriteLine(salesTotal);
// and request money for the order
Console.Write("Amount Tended? ");
amountTended = double.Parse(Console.ReadLine());
// Calculate the difference owed to the customer
// or that the customer still owes to the store
moneyChange = amountTended - salesTotal;
Console.Clear();
// Display the receipt
Console.WriteLine("=====================================");
Console.WriteLine("-/- Georgetown Dry Cleaning Services -/-");
Console.WriteLine("=====================================");
Console.Write("Customer:  ");
Console.WriteLine(customerName);
Console.Write("Home Phone: ");
Console.WriteLine(homePhone);
Console.Write("Date & Time: ");
Console.WriteLine(orderDate);
Console.WriteLine("-------------------------------");
Console.WriteLine("Item Type  Qty Unit/Price Sub-Total");
Console.WriteLine("-------------------------------");
Console.Write("Shirts   ");
Console.Write(numberOfShirts);
Console.Write("   ");
Console.Write(PriceOneShirt);
Console.Write("   ");
Console.WriteLine(subTotalShirts);
Console.Write("Pants    ");
Console.Write(numberOfPants);
Console.Write("   ");
Console.Write(PriceAPairOfPants);
Console.Write("   ");
Console.WriteLine(subTotalPants);
Console.Write("Dresses   ");
Console.Write(numberOfDresses);
Console.Write("   ");
Console.Write(PriceOneDress);
```

```
        Console.Write("    ");
        Console.WriteLine(subTotalDresses);
        Console.WriteLine("-----------------------------------");
        Console.Write("Total Order:    ");
        Console.WriteLine(totalOrder);
        Console.Write("Tax Rate:      ");
        Console.Write(TaxRate * 100);
        Console.WriteLine('%');
        Console.Write("Tax Amount:    ");
        Console.WriteLine(taxAmount);
        Console.Write("Net Price:     ");
        Console.WriteLine(salesTotal);
        Console.WriteLine("-----------------------------------");
        Console.Write("Amount Tended:  ");
        Console.WriteLine(amountTended);
        Console.Write("Difference:    ");
        Console.WriteLine(moneyChange);
        Console.WriteLine("=====================================");
        System.Console.ReadKey();
        return 0;
    }
  }
```

۲. حال برنامه را اجرا کنید . نتیجه ی زیر به دست می آید.

```
-/- Georgetown Dry Cleaning Services -/-
    Enter Customer Name: Alexander Pappas
    Enter Customer Phone: (301) 397-9764
    Enter the order date and time (mm/dd/yyyyhh:mm AM/PM)
    06/22/98 08:26 AM
    Number of Shirts: 2
    Number of Pants: 6
    Number of Dresses: 0
    The Total order is: 20.727000
    Amount Tended? 50
```

۳. کلید Enter را فشار دهید.

```
=====================================
-/- Georgetown Dry Cleaning Services -/-
=====================================
Customer:  Alexander Pappas Home Phone: (301) 397-9764
 Date & Time: 6/22/1998 8:26:00 AM
 -----------------------------------
 Item Type  Qty Unit/Price Sub-Total
 -----------------------------------
```

```
Shirts    2   0.95   1.90
Pants     6   2.95   17.70
Dresses   0   4.55   0
-------------------------------------
Total Order:    19.60
Tax Rate:       5.7500%
Tax Amount:     1.127000
Net Price:      20.727000
-------------------------------------
Amount Tended:  50
Difference:     29.273000
=====================================
```

۴. به محیط برنامه نویسی برگردید.

نمایش داده ها با placeholder

۱. برای این منظور فایل مورد نظر را به صورت زیر اصلاح کنید.

```
using System;
  public class OrderProcessing
  {
    public static int Main()
    {
      // Price of items
      const double PriceOneShirt = 0.95;
      const double PriceAPairOfPants = 2.95;
      const double PriceOneDress = 4.55;
      const double TaxRate = 0.0575;  // 5.75%
      double Difference;
      // Basic information about an order
      string customerName, homePhone;
      DateTime orderDate;
      // Unsigned numbers to represent cleaning items
      uint numberOfShirts, numberOfPants, numberOfDresses;
      // Each of these sub totals will be used for cleaning items
      double subTotalShirts, subTotalPants, subTotalDresses;
      // Values used to process an order
      double totalOrder, taxAmount, salesTotal;
      double amountTended, moneyChange;
      Console.Title = "Georgetown Dry Cleaning Services";
      Console.WriteLine("-/- Georgetown Dry Cleaning Services -/-");
      // Request order information from the user
      Console.Write("Enter Customer Name: ");
      customerName = Console.ReadLine();
      Console.Write("Enter Customer Phone: ");
      homePhone = Console.ReadLine();
      Console.WriteLine("Enter the order date and " +"time (mm/dd/yyyy hh:mm AM/PM)");
```

```
orderDate = DateTime.Parse(Console.ReadLine());
// Request the quantity of each category of items
Console.Write("Number of Shirts: ");
numberOfShirts = uint.Parse(Console.ReadLine());
Console.Write("Number of Pants: ");
numberOfPants = uint.Parse(Console.ReadLine());
Console.Write("Number of Dresses: ");
numberOfDresses = uint.Parse(Console.ReadLine());
// Perform the necessary calculations
subTotalShirts = numberOfShirts * PriceOneShirt;
subTotalPants = numberOfPants * PriceAPairOfPants;
subTotalDresses = numberOfDresses * PriceOneDress;
// Calculate the "temporary" total of the order
totalOrder = subTotalShirts +subTotalPants + subTotalDresses;
// Calculate the tax amount using a constant rate
taxAmount = totalOrder * TaxRate;
// Add the tax amount to the total order
salesTotal = totalOrder + taxAmount;
// Communicate the total to the user…
Console.Write("\nThe Total order is: ");
Console.WriteLine(salesTotal);
// and request money for the order
Console.Write("Amount Tended? ");
amountTended = double.Parse(Console.ReadLine());
// Calculate the difference owed to the customer
// or that the customer still owes to the store
Difference = amountTended - salesTotal;
Console.Clear();
// Display the receipt
Console.WriteLine("=====================================");
Console.WriteLine("-/- Georgetown Dry Cleaning Services -/-");
Console.WriteLine("=====================================");
Console.WriteLine("Customer:  {0}", customerName);
Console.WriteLine("Home Phone: {0}", homePhone);
Console.WriteLine("Date & Time: {0}", orderDate);
Console.WriteLine("-------------------------------------");
Console.WriteLine("Item Type  Qty Unit/Price Sub-Total");
Console.WriteLine("-------------------------------------");
Console.WriteLine("Shirts    {0}   {1}   {2}",numberOfShirts,PriceOneShirt, subTotalShirts);
Console.WriteLine("Pants     {0}   {1}   {2}", numberOfPants,PriceAPairOfPants, subTotalPants);
Console.WriteLine("Dresses   {0}   {1}   {2}",numberOfDresses, PriceOneDress, subTotalDresses);
Console.WriteLine("-------------------------------------");
Console.WriteLine("Total Order:  {0}", totalOrder);
Console.WriteLine("Tax Rate:     {0}%", TaxRate * 100);
Console.WriteLine("Tax Amount:   {0}", taxAmount);
Console.WriteLine("Net Price:    {0}", salesTotal);
Console.WriteLine("-------------------------------------");
Console.WriteLine("Amount Tended: {0}", amountTended);
Console.WriteLine("Difference:   {0}", Difference);
Console.WriteLine("=====================================");
```

```
        System.Console.ReadKey();
        return 0;
    }
  }
}
```

۲. برنامه را اجرا کرده تا نتیجه ی آن را مشاهده کنید.

۳. اکنون پنجره ی DOS را ببندید

تبدیل به رشته

پیش تر ذکر شد که هر آنچه در محیط C# با صفحه کلید تایپ می شود عمدتاً رشته است و تبدیل آن به نوع مورد نیاز، دیگر وظیفه ی برنامه نویس می باشد و برعکس، چنانچه مقداری داشته باشید که (در قالب) رشته نبود می توان به راحتی آن را به رشته تبدیل کرد. برای پشتیبانی از چنین امکان و قابلیتی، تمامی نوع داده های.NET Framework مکانیسمی به نام ToString را ارائه می دهند. در زبان C# پروسه ی تبدیل معمولاً به صورت اتوماتیک توسط compiler انجام می شود. با این وجود، در برخی موارد شما خود مجبور به انجام فرایند تبدیل هستید.

به منظور تبدیل مقداری از داده های نوع اولیه به رشته، اسم متغیر و به دنبال آن عملگر نقطه و تابع ToString() را تایپ کنید. توجه خود را به مثال زیر جلب کنید.

```
using System;
public class Exercise
{
  public static void Main()
  {
    var fullName = "Anselme Bogos";
    var age = 15;
    var hSalary = 22.74;
    Console.WriteLine("Full Name: {0}", fullName);
    Console.WriteLine("Age: {0}", age.ToString());
    Console.WriteLine("Distance: {0}", hSalary.ToString());
    Console.WriteLine();
  }
}
```

در برخی حالات می توان داده داخل پرانتز تابع ToString() قرار داد.

قالب بندی نمایش داده ها

۱. برای این منظور، فایل را به ترتیب زیر تغییر دهید.

```
using System;
public class OrderProcessing
{
  public static int Main()
  {
    // Price of items
    const double PriceOneShirt = 0.95;
    const double PriceAPairOfPants = 2.95;
    const double PriceOneDress = 4.55;
    const double TaxRate = 0.0575;  // 5.75%
    // Basic information about an order
    string customerName, homePhone;
    DateTime orderDate;
    // Unsigned numbers to represent cleaning items
    uint numberOfShirts, numberOfPants, numberOfDresses;
    // Each of these sub totals will be used for cleaning items
    double subTotalShirts, subTotalPants, subTotalDresses;
    // Values used to process an order
    double totalOrder, taxAmount, salesTotal;
    double amountTended,Difference, moneyChange;
    Console.Title = "Georgetown Dry Cleaning Services";
    Console.WriteLine("-/- Georgetown Dry Cleaning Services -/-");
    // Request order information from the user
    Console.Write("Enter Customer Name: ");
    customerName = Console.ReadLine();
    Console.Write("Enter Customer Phone: ");
    homePhone = Console.ReadLine();
    Console.WriteLine("Enter the order date and " +"time (mm/dd/yyyy hh:mm AM/PM)");
    orderDate = DateTime.Parse(Console.ReadLine());
    // Request the quantity of each category of items
    Console.Write("Number of Shirts: ");
    numberOfShirts = uint.Parse(Console.ReadLine());
    Console.Write("Number of Pants:  ");
    numberOfPants = uint.Parse(Console.ReadLine());
    Console.Write("Number of Dresses: ");
    numberOfDresses = uint.Parse(Console.ReadLine());
    // Perform the necessary calculations
    subTotalShirts = numberOfShirts * PriceOneShirt;
    subTotalPants = numberOfPants * PriceAPairOfPants;
    subTotalDresses = numberOfDresses * PriceOneDress;
    // Calculate the "temporary" total of the order
    totalOrder = subTotalShirts + subTotalPants + subTotalDresses;
    // Calculate the tax amount using a constant rate
    taxAmount = totalOrder * TaxRate;
    // Add the tax amount to the total order
    salesTotal = totalOrder + taxAmount;
    // Communicate the total to the user…
    Console.Write("\nThe Total order is: ");
    Console.WriteLine(salesTotal);
    // and request money for the order
```

```
        Console.Write("Amount Tended? ");
        amountTended = double.Parse(Console.ReadLine());
        // Calculate the difference owed to the customer
        // or that the customer still owes to the store
        Difference = amountTended - salesTotal;
        Console.Clear();
        // Display the receipt
        Console.WriteLine("=====================================");
        Console.WriteLine("-/- Georgetown Dry Cleaning Services -/-");
        Console.WriteLine("=====================================");
        Console.WriteLine("Customer:  {0}", customerName);
        Console.WriteLine("Home Phone: {0}", homePhone);
        Console.WriteLine("Date & Time: {0}", orderDate);
        Console.WriteLine("-----------------------------------");
        Console.WriteLine("Item Type  Qty Unit/Price Sub-Total");
        Console.WriteLine("-----------------------------------");
        Console.WriteLine("Shirts    {0}   {1:C}   {2}",numberOfShirts.ToString(),
        PriceOneShirt,subTotalShirts.ToString("C"));
        Console.WriteLine("Pants     {0}   {1:C}   {2:C}", numberOfPants, PriceAPairOfPants,subTotalPants);
        Console.WriteLine("Dresses   {0}   {1:C}   {2:C}", numberOfDresses, PriceOneDress,subTotalDresses);
        Console.WriteLine("-----------------------------------");
        Console.WriteLine("Total Order:  {0:C}", totalOrder);
        Console.WriteLine("Tax Rate:     {0:P}", TaxRate);
        Console.WriteLine("Tax Amount:   {0}",taxAmount.ToString("C"));
        Console.WriteLine("Net Price:    {0:F}", salesTotal);
        Console.WriteLine("-----------------------------------");
        Console.WriteLine("Amount Tended: {0:C}", amountTended);
        Console.WriteLine("Difference:   {0:C}", Difference);
        Console.WriteLine("=====================================");
        System.Console.ReadKey();
        return 0;
    }
}
```

۲. برنامه را اجرا کنید.

```
-/- Georgetown Dry Cleaning Services -/-
Enter Customer Name: Gretchen McCormack
Enter Customer Phone: (410) 739-2884
Enter the order date and time (mm/dd/yyyyhh:mm AM/PM)
04/09/2001 10:25 AM
Number of Shirts: 5
Number of Pants:  12
Number of Dresses: 8
The Total order is: 80.951625
Amount Tended? 100
```

۳. حال Enter را بزنید.

```
=====================================
-/- Georgetown Dry Cleaning Services -/-
```

```
====================================
Customer:  Gretchen McCormack
Home Phone: (410) 739-2884
Date & Time: 4/9/2001 10:25:00 AM
------------------------------------
Item Type  Qty Unit/Price Sub-Total
------------------------------------
Shirts    5   $0.95   $4.75
Pants     12   $2.95   $35.40
Dresses   8   $4.55   $36.40
------------------------------------
Total Order:   $76.55
Tax Rate:     5.75 %
Tax Amount:    $4.40
Net Price:    80.95
------------------------------------
Amount Tended: $100.00
Difference:    $19.05
====================================
```

۴. پنجره ی DOS را ببندید.

قالب بندی خط

در برنامه های فوق به منظور نمایش خطی از یک متن، از توابع Write() یا WriteLine() استفاده می کردیم. برای قرار دادن متنی با طول متفاوت بالای متن دیگری، با استفاده ی بیش از حد از فضاهای خالی، یک رشته را خراب می کردیم. چنین تکنیکی تاحدی غیر حرفه ای و نامطمئن تلقی می شود. خوشبختانه، قالب بندی خط و متن، همچنین نحوه ی نمایش آن ها را می توان تاحدی زیادی مدیریت کرد. همان طور که انتظار می رفت، .NET Framework به منظور تنظیم مقدار فضای مورد نیاز برای نمایش رشته ای از متن و نحوه ی قرار گیری رشته ی مزبور در خط مربوط به آن مکانیسم های ویژه ای در اختیار کاربر قرار می دهد.

به منظور تعیین مقدار فضای مورد نیاز برای نمایش رشته، می توان از placeholder داخل پرانتز Write() یا WriteLine() استفاده کرد. برای این منظور، داخل placeholder، ۰ یا تعدادی عدد تصاعدی (incrementing number) را به همراه کاراکتر قالب بندی آن (formatting character) تایپ کنید. سپس، ویرگول و به دنبال آن تعداد کاراکترهایی که با فضای (پهنا) دلخواه همخوانی دارد / برابر است را وارد کنید.

مثال

```
using System;
public class Exercise
{
```

```
    public static void Main()
    {
      var fullName = "Anselme Bogos";
      var age = 15;
      var hSalary = 22.74;
      Console.WriteLine("Full Name: {0,20}", fullName);
      Console.WriteLine("Age:{0,14}", age.ToString());
      Console.WriteLine("Distance: {0:C,8}", hSalary.ToString());
      Console.WriteLine();
    }
  }
```

نتیجه

```
Full Name:      Anselme Bogos
Age:        15
Distance: 22.74
```

علامتی که برای پهنا (width) در نظر می گیرید بسیار مهم است. اگر مثبت بود، خط متن در سمت راست قرار می گیرد که باید alignment مرجع، به خصوص برای مقادیر عددی باشد. حال چنانچه علامتی که ارائه می دهید منفی باشد، متن در سمت چپ قرار می گیرد.

قالب بندی تاریخ و زمان

همان طور که پیش تر ذکر شد، هنگامی که کاربر مقدار زمان برای متغیر DateTime وارد می کند، compiler خود یک بخش زمان به مقدار مذکور اضافه می کند. اگر بخواهید تنها بخش زمان یا بخش تاریخ بررسی شود، لازم است این امر را برای compiler مشخص کنید. برای این منظور، نوع داده ی DateTime یک سری حروف در نظر گرفته است که با استفاده از آن می توان نحوه ی نمایش مقدار (نوع قالب بندی آن) برای کاربر را تعیین کرد. کاراکتر مورد نظر داخل placeholder متغیر DateTime پس از ۰ یا مقدار عددی افزایشی (incremental numeric value) باید تایپ شود.

مدیریت قالب بندی تاریخ / زمان

۱. برای تنظیم قالب بندی زمان و تاریخ، فایل را به صورت زیر اصلاح کنید.

```
using System;
  public class OrderProccessing
  {
    public static int Main()
    {
      //Price of items
      const double PriceOneShirt = 0.95;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
const double PriceAPairOfPants = 2.95;
const double PriceOneDress = 4.55;
const double TaxRate = 0.0575;  //5.75%
//Basic information about an order
string customerName, homephone;
DateTime orderDate, orderTime;
//Unsigned numbers to represent cleaning items
uint numberOfShirts, numberOfPants, numberOfDresses;
//Each of these sub totals will be used for cleaning items
double subTotalShirts, subTotalPants, subTotalDresses;
//Values used to process an order
double totalOrder, taxAmount, salesTotal;
double amountTended, moneyChange;
Console.Title = "Georgetown Dry Cleaning Services";
Console.WriteLine("-/- Georgetown Dry Cleaning Services -/-");
//Request order information from the user
Console.Write("Enter Customer Name: ");
customerName = Console.ReadKey().ToString();
Console.Write("Enter Customer Phone: ");
homephone = Console.ReadLine();
Console.Write("Enter the order date(mm/dd/yyyy): ");
orderDate = DateTime.Parse(Console.ReadLine());
Console.Write("Enter the order time(hh:mm AM/PM): ");
orderTime = DateTime.Parse(Console.ReadLine());
//Request the quantity of each category of items
Console.Write("Number of Shirts: ");
numberOfShirts = uint.Parse(Console.ReadLine());
Console.Write("Number of Pants: ");
numberOfPants = uint.Parse(Console.ReadLine());
Console.Write("Number of Dresses: ");
numberOfDresses = uint.Parse(Console.ReadLine());
//Perform the necessary calculations
subTotalShirts = numberOfShirts * PriceOneShirt;
subTotalPants = numberOfPants * PriceAPairOfPants;
subTotalDresses = numberOfDresses * PriceOneDress;
//Calculate the "temporary" total of the order
totalOrder = subTotalShirts + subTotalPants +subTotalDresses;
//Calculate the tax amount using a constant rate
taxAmount = totalOrder * TaxRate;
//Add the tax amount to the total order
salesTotal = totalOrder + taxAmount;
//Communicate the total to the user…
Console.WriteLine("\nThe Total order is: {0:C}", salesTotal);
//and request money for the order
Console.Write("Amount Tended? ");
amountTended = double.Parse(Console.ReadLine());
//Calculate the difference owed to the customer
//or that the customer still owes to the store
moneyChange = amountTended - salesTotal;
Console.Clear();
```

```
        //Display the receipt
        Console.WriteLine("=====================================");
        Console.WriteLine("-/- Georgetown Dry Cleaning Services -/-");
        Console.WriteLine("=====================================");
        Console.WriteLine("Customer:  {0}", customerName);
        Console.WriteLine("Home Phone:{0}", homephone);
        Console.WriteLine("Order Date:{0:D}", orderDate);
        Console.WriteLine("Order Time:{0:t}", orderTime);
        Console.WriteLine("-----------------------------------");
        Console.WriteLine("Item Type  Qty Unit/Price Sub-Total");
        Console.WriteLine("-----------------------------------");
        Console.WriteLine("Shirts   {0}  {1}    {2}",numberOfShirts.ToString(),PriceOneShirt.ToString("C"), subTotalShirts.ToString("C"));
        Console.WriteLine("Pants    {0}  {1}    {2}",numberOfPants.ToString(),PriceAPairOfPants.ToString("C"),subTotalPants.ToString("C"));
        Console.WriteLine("Dresses  {0}  {1}    {2}", numberOfDresses.ToString(),PriceOneDress.ToString("C"),subTotalDresses.ToString("C"));
        Console.WriteLine("-----------------------------------");
        Console.WriteLine("Total Order:  {0}",totalOrder.ToString("C"));
        Console.WriteLine("Tax Rate:    {0}",TaxRate.ToString("P"));
        Console.WriteLine("Tax Amount:  {0}", taxAmount.ToString("C"));
        Console.WriteLine("Net Price:   {0}",  salesTotal.ToString("C"));
        Console.WriteLine("-----------------------------------");
        Console.WriteLine("Amount Tended:{0}", amountTended.ToString("C"));
        Console.WriteLine("Difference:  {0}",moneyChange.ToString("C"));
        Console.WriteLine("=====================================");
        System.Console.ReadKey();
        return 0;
    }
  }
}
```

۲. برنامه را اجرا کنید

```
-/- Georgetown Dry Cleaning Services -/-
Enter Customer Name: Antoinette Calhoun
Enter Customer Phone: (703) 797-1135
Enter the order date(mm/dd/yyyy): 04/12/2002
Enter the order time(hh:mm AM/PM): 2:12 PM
Number of Shirts: 5
Number of Pants:  2
Number of Dresses: 1
The Total order is: $16.07
Amount Tended? 20
```

۳. اکنون کلید Enter را فشار دهید.

```
=====================================
/- Georgetown Dry Cleaning Services -/-
=====================================
ustomer:  Antoinette Calhoun
ome Phone: (703) 797-1135
rder Date: Friday, April 12, 2002
rder Time: 2:12 PM
```

```
-----------------------------------
tem Type  Qty Unit/Price Sub-Total
-----------------------------------
hirts    5  $0.95     $4.75
ants     2  $2.95     $5.90
Dresses  1  $4.55     $4.55
-----------------------------------
Total Order:  $15.20
Tax Rate:     5.75 %
Tax Amount:   $0.87
Net Price:    $16.07
-----------------------------------
Amount Tended: $20.00
Difference:   $3.93
===================================
```

۴. پنجره ی DOS را بسته و به محیط برنامه نویسی برگردید.

۵. به فهرست اصلی مراجعه کرده، گزینه ی File -> Close Solution را کلیک کنید.

۶. زمانی که از شما درخواست شد می خواهید Save کنید یا نه روی no کلیک کنید.

ترکیبات کلاس ها

تودرتو کردن کلاس ها (class nesting)

می توان کلاسی را در دل کلاسی دیگر گنجاند. به کلاسی که داخل کلاسی دیگر ایجاد می شود، کلاس nested یا تودرتو می گویند. برای این منظور

داخل کلاس موجود کلیک کرده و کد مورد نیاز را برای کلاس جدید وارد کنید، ابتدا کلید واژه ی class، سپس یک اسم و کاراکتر {} را تایپ کنید.

حال، کل کلاس را انتخاب کنید. روی انتخاب راست کلیک کرده و گزینه ی Surround With... را انتخاب کنید. در لیستی که ظاهر می شود روی class دوبار کلیک کنید.

در مثال زیر کلاسی به نام Inside داخل کلاس دیگری به نام Outside قرار گرفته

```
public class Outside
{
  public class Inside
  {
  }
}
```

طبق دستور بالا می توان هر تعداد کلاس که لازم است در کلاس های دیگر جای گذاری کرد. طریقه ی مدیریت و کنترل کلاس تودرتو هیچ تفاوتی با (مدیریت) یک کلاس عادی ندارد. برای مثال، می توان تمامی فیلدها، متدها و properties مورد نیاز را داخل کلاس تودرتو یا کلاس میزبان (کلاس تودرتو) تعریف کرد. هنگامی که کلاسی را در کلاس دیگر جای گذاری می کنید، هیچ گونه ارتباط (برنامه ای) خاصی بین دو کلاس به وجود نمی آید : یعنی تنها به صرف قرار گرفتن یک کلاس در کلاس دیگر، کلاس تودرتو (nested class یا کلاس گنجانده شده) به اعضای کلاس میزبان (nesting class) دسترسی فوری پیدا نخواهد کرد. در حقیقت، این دو کلاس از هم مجزا هستند.

اسم کلاس تودرتو (کلاس گنجانده شده) بیرون از کلاس میزبان قابل رویت نیست. برای دسترسی به کلاس تودرتو بیرون از کلاس میزبان، باید اسم کلاس تودرتو را هر جایی که می خواهید از آن استفاده کنید، تعریف کنید. اگر می خواهید متغیر Inside را جایی درون برنامه ولی بیرون از Outside تعریف کنید، لازم است اسم آن را تعریف کنید.

مثال

```
using System;
public class Outside
{
  public class Inside
  {
    public Inside()
    {
      Console.WriteLine(" -= Inside =-");
    }
  }
  public Outside()
  {
    Console.WriteLine(" =- Outside -=");
  }
}
public class Exercise
{
  static int Main()
  {
    Outside recto = new Outside();
    Outside.Inside ins = new Outside.Inside();
    return 0;
  }
}
```

نتیجه ی زیر را به دست می دهد.

```
=- Outside -=
 -= Inside =-
```

به این خاطر که هیچ گونه ارتباط خاصی بین کلاس گنجانده شده و کلاس میزبان (دربردارنده) وجود ندارد، برای دسترسی به کلاس گنجانده شده در کلاس میزبان، باید از اعضای ایستا (static members) آن استفاده کرد. به عبارت دیگر، می توانید همه ی اعضای کلاس گنجانده شده را که می خواهید در کلاس میزبان دسترسی داشته باشید به عنوان ایستا تعریف کنید. به مثال زیر توجه کنید.

```
using System;
public class Outside
{
  public class Inside
  {
    public static string InMessage;
    public Inside()
    {
      Console.WriteLine(" -= Insider =-");
      InMessage = "Sitting inside while it's raining";
    }
    public static void Show()
    {
      Console.WriteLine("Show me the wonderful world of C# Programming");
    }
  }
  public Outside()
  {
    Console.WriteLine(" =- The Parent -=");
  }
  public void Display()
  {
    Console.WriteLine(Inside.InMessage);
    Inside.Show();
  }
}
class Exercise
{
  static int Main()
  {
    Outside recto = new Outside();
    Outside.Inside ins = new Outside.Inside();
    recto.Display();
    return 0;
  }
}
```

به همین نحو، برای دسترسی به کلاس میزبان در کلاس گنجانده شده، می توان از اعضای ایستا کلاس میزبان استفاده کرد (کلیه ی اعضای کلاس میزبان را که می خواهید در کلاس گنجانده شده دسترسی داشته باشید به عنوان ایستا تعریف کنید).

مثال

```
using System;
  public class Outside
  {
    public class Inside
    {
      public static string InMessage;
      public Inside()
      {
        Console.WriteLine(" -= Insider =-");
        InMessage = "Sitting inside while it's raining";
      }
      public static void Show()
      {
        Console.WriteLine("Show me the wonderful world of C# Programming");
      }
      public void FieldFromOutside()
      {
        Console.WriteLine(Outside.OutMessage);
      }
    }
    private static string OutMessage;
    public Outside()
    {
      Console.WriteLine(" =- The Parent -=");
      OutMessage = "Standing outside! It's cold and raining!!";
    }
    public void Display()
    {
      Console.WriteLine(Inside.InMessage);
      Inside.Show();
    }
  }
  public class Exercise
  {
    static int Main()
    {
      Outside recto = new Outside();
      Outside.Inside Ins = new Outside.Inside();
      recto.Display();
      Console.WriteLine();
      Ins.FieldFromOutside();
      return 0;
    }
  }
```

نتیجه ی زیر به دست می آید.

```
 =- The Parent -=

 -= Insider =-
```

Sitting inside while it's raining

Show me the wonderful world of C# Programming

Standing outside! It's cold and raining!!

همچنین می توان به جای اعضای ایستا، اگر می خواهید به اعضای کلاس گنجانده شده در کلاس میزبان دسترسی داشته باشید، متغیر کلاس گنجانده شده را در کلاس میزبان تعریف کنید و بر عکس (عکس این قضیه را برای دسترسی به اعضای کلاس میزبان در کلاس گنجانده شده پیاده کنید).

مثال

```
using System;
  public class Outside
  {
    // A member of the nesting class
    private string OutMessage;
    // The nested class
    public class Inside
    {
      // A field in the nested class
      public string InMessage;
      // A constructor of the nested class
      public Inside()
      {
        Console.WriteLine(" -= Insider =-");
        this.InMessage = "Sitting inside while it's raining";
      }
      // A method of the nested class
      public void Show()
      {
        // Declare a variable to access the nesting class
        Outside outsider = new Outside();
        Console.WriteLine(outsider.OutMessage);
      }
    } // End of the nested class
    // A constructor of the nesting class
    public Outside()
    {
      this.OutMessage = "Standing outside! It's cold and raining!!";
      Console.WriteLine(" =- The Parent -=");
    }
    // A method of the nesting class
    public void Display()
    {
```

```
        Console.WriteLine(insider.InMessage);
    }
    // Declare a variable to access the nested class
    Inside insider = new Inside();
}
public class Exercise
{
    static int Main()
    {
        Outside recto = new Outside();
        Outside.Inside Ins = new Outside.Inside();
        Ins.Show();
        recto.Display();
        return 0;
    }
}
```

نتیجه ی زیر به دست می آید.

```
-= Insider =-
 =- The Parent -=
 -= Insider =-
 -= Insider =-
 =- The Parent -=
Standing outside! It's cold and raining!!
Sitting inside while it's raining
```

کلاسی به عنوان فیلد

مثل تمامی متغیرهایی که تاکنون به کار برده ایم، می توان یک کلاس یا ساختار را متغیر عضو کلاسی دیگر کرد. برای این که بتوانید کلاس دیگری را در کلاس خود به کار ببرید، در وهله ی اول لازم است آن کلاس را در دسترس داشته باشید. هم می توانید از کلاس های آماده ی C# برای این منظور استفاده کنید، هم می توانید کلاس خود را ایجاد کنید.

مثال

```
public class Point
{
    internal short x;
    internal short y;
}
```

فیلد در حقیقت متغیر عضوی است که به جای نوع اولیه (primitive type) از کلاس دیگری ساخته می شود. برای استفاده از یک کلاس به عنوان متغیر عضو کلاسی دیگر، کافی است متغیر آن را تعریف کنید. به مثال زیر توجه کنید.

```
public class Point
{
  internal short x;
  internal short y;
}
public class CoordinateSystem
{
  public Point start;
}
```

پس از تعریف کلاسی به عنوان عضو متغیر کلاس دیگر، می توان از متغیر مزبور به طور منظم استفاده کرد. به این خاطر که عضو مورد نظر یک کلاس است، که به عنوان (یک) ارجاع (reference) معرفی شده، قوانینی وجود دارد که هنگام استفاده از آن باید رعایت کرد. پس از تعریف متغیر عضو، باید مطمئن شوید که حافظه ی کافی به آن تخصیص داده شده است. همچنین لازم است متغیر مذکور صحیح و به طور مقتضی مقداردهی (اولیه) شده باشد. در غیر این صورت، هنگام ترجمه ی (compile) برنامه با error مواجه می شوید.

استفاده از کلاس به عنوان فیلد

۱. برنامه ی Microsoft Visual C# را راه اندازی کنید.

۲. برای ایجاد برنامه ی جدید، در فهرست اصلی برنامه روی File -> New Project... کلیک کنید.

۳. در لیست میانی، روی گزینه ی Empty Project... کلیک کنید.

۴. اسم پروژه را به ElectonicStore1 تغییر داده، حال کلید Enter را فشار دهید.

۵. به منظور ایجاد کلاس جدید، در پنجره ی Solution Explorer، روی اسم پروژه راست کلیک کرده، اکنون نشانگر موس را روی Add قرار دهید و گزینه ی class را انتخاب کنید.

۶. اسم کلاس را StoreItem انتخاب کنید، سپس روی گزینه ی Add کلیک کنید.

۷. فایل را به صورت زیر تکمیل کنید.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ElectronicStore1
{
  public class StoreItem
  {
    private long nbr;
    private char cat;
    private string mk;
    private string mdl;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    private double price;
    public long GetItemNumber()
    {
      return nbr;
    }
    public void SetItemNumber(long number = 0)
    {
      this.nbr = number;
    }
    public char GetCategory()
    {
      return cat;
    }
    public void SetCategory(char category = 'A')
    {
      this.cat = category;
    }
    public string GetMake()
    {
      return mk;
    }
    public void SetMake(string make = "Unknown")
    {
      this.mk = make;
    }
    public string GetModel()
    {
      return mdl;
    }
    public void SetModel(string model = "Unknown")
    {
      this.mdl = model;
    }
    public double GetUnitPrice()
    {
      return price;
    }
    public void SetUnitPrice(double unitPrice = 0.00D)
    {
      this.price = unitPrice;
    }
  }
}
```

۸. برای ایجاد فایل جدید، فهرست گزینه ی اصلی را باز کرده، Project -> Add New Item... را انتخاب کنید.

۹. گزینه ی Code File را از لیست میانی انتخاب کنید.

۱۰. اسم را به OrderProcessing تغییر داده و کلید Enter را فشار دهید.

۱۱. فایل را به ترتیب زیر اصلاح کنید.

```
using System;
 namespace ElectronicStore1
{
   public class OrderProcessing
   {
     public static int Main()
     {
       string strTitle1 = "=-= Nearson Electonics =-=\n";
       string strTitle2 = "******* Store Items ******";
       System.Console.ReadKey();
       return 0;
     }
   }
}
```

کلاس به عنوان یک نوع

بازگرداندن شی از متد

می توان مقدار یک کلاس را از متد یک کلاس بازگرداند. برای این منظور، ابتدا متد را تعریف کرده، سپس کلاس را به عنوان نوع بازگشتی (return type) مشخص می کنید.

مثال

```
public class Point
{
   internal short x;
   internal short y;
}
public class CoordinateSystem
{
   private Point start;
   private Point end;
   public Point GetThePoint()
   {
   }
}
```

پس از پیاده سازی (implementing) متد، باید مقداری بازگردانید که با کلاس همخوانی داشته باشد، در غیر این صورت حین ترجمه / کامپایل برنامه با error مواجه می شوید. بنابراین، ابتدا متغیر کلاس را در بدنه ی متد تعریف می کنید، در این خلال متغیر را مقداردهی اولیه می کنید، سپس آن را باز می گردانید.

مثال

```
public class Point
{
  internal short x;
  internal short y;
}
public class CoordinateSystem
{
  private Point start;
  private Point end;
  public Point GetThePoint()
  {
    Point pt = new Point();
    Console.Write("Enter the x coordinate of the point: ");
    pt.x = short.Parse(Console.ReadLine());
    Console.Write("Enter the y coordinate of the point: ");
    pt.y = short.Parse(Console.ReadLine());
    return pt;
  }
}
```

پس از این که متد مقدار کلاس را بازگرداند، می توانید آن را طبق نیاز خود مورد استفاده قرار دهید.

ارسال کلاس به عنوان آرگومان

پس از ایجاد کلاس، می توان آن را مثل هر متغیر دیگری به کار برد. برای مثال، می توان متغیر آن را به عنوان آرگومان به متد کلاس دیگری ارسال کرد. زمانی که کلاسی به عنوان آرگومان ارسال می شود، اعضای عمومی (public members) کلاس در اختیار متدی که از آن استفاده می کند قرار می گیرد. همچنین می توان چندین (بیش از یک) کلاس به عنوان آرگومان به یک متد ارسال کرد.

مثال

```
using System;
  namespace Geometry
  {
    public class Point
    {
      internal short x;
      internal short y;
    }
    public class CoordinateSystem
    {
      public Point start;
      public Point end;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
  public Point GetThePoint()
  {
    Point pt = new Point();
    Console.Write("Enter the x coordinate of the point: ");
    pt.x = short.Parse(Console.ReadLine());
    Console.Write("Enter the y coordinate of the point: ");
    pt.y = short.Parse(Console.ReadLine());
    return pt;
  }
  public double DistanceFromOrigin(Point pt)
  {
    double sqr1 = Math.Pow(pt.x, 2);
    double sqr2 = Math.Pow(pt.y, 2);
    double distance = Math.Sqrt(sqr1 + sqr2);
    return distance;
  }
  public double DistanceBetween2Points(Point pt1, Point pt2)
  {
    double sqr1 = Math.Pow(pt2.x - pt1.x, 2);
    double sqr2 = Math.Pow(pt2.y - pt1.y, 2);
    double distance = Math.Sqrt(sqr1 + sqr2);
    return distance;
  }
}
public class Program
{
  private static CoordinateSystem IdentifyCoordinates()
  {
    CoordinateSystem coord = new CoordinateSystem();
    Console.WriteLine("Start Point");
    coord.start = coord.GetThePoint();
    Console.WriteLine("End Point");
    coord.end = coord.GetThePoint();
    return coord;
  }
  private static void Show(CoordinateSystem c)
  {
    Console.WriteLine("Coordinate System");
    Console.WriteLine("Starting Point: P({0}, {1})", c.start.x, c.start.y);
    Console.WriteLine("Ending Point:  Q({0}, {1})", c.end.x, c.end.y);
    Console.WriteLine("Distance Between Both Points: {0:F}",c.DistanceBetween2Points(c.start, c.end));
  }
  static int Main()
  {
    CoordinateSystem coord = IdentifyCoordinates();
    Console.WriteLine();
    Show(coord);
    return 0;
  }
}
```

```
}
```

زیر نمونه ای از (اجرای) برنامه مورد نظر را مشاهده می کنید.

```
Start Point
Enter the x coordinate of the point: -2
Enter the y coordinate of the point: 2
End Point
Enter the x coordinate of the point: 3
Enter the y coordinate of the point: -6
Coordinate System
Starting Point: P(-2, 2)
Ending Point:  Q(3, -6)
Distance Between Both Points: 9.43
Press any key to continue...
```

به این خاطر که کلاس ها همیشه به عنوان ارجاع (reference) مورد استفاده قرار می گیرند، هنگام ارسال کلاس به عنوان آرگومان، کلاس نام برده (به صورت ضمنی) با ارجاع فرستاده می شود. برای این منظور می توانید کلیدواژه ی ref را در سمت چپ آرگومان تایپ کنید.

مثال

```
using System;
namespace ConsoleApplication1
{
  public class Point
  {
    internal short x;
    internal short y;
  }
  public class CoordinateSystem
  {
    public Point start;
    public Point end;
    public Point GetThePoint()
    {
      Point pt = new Point();
      Console.Write("Enter the x coordinate of the point: ");
      pt.x = short.Parse(Console.ReadLine());
      Console.Write("Enter the y coordinate of the point: ");
      pt.y = short.Parse(Console.ReadLine());
      return pt;
    }
    public double DistanceFromOrigin(ref Point pt)
    {
      double sqr1 = Math.Pow(pt.x, 2);
      double sqr2 = Math.Pow(pt.y, 2);
```

```
        double distance = Math.Sqrt(sqr1 + sqr2);
        return distance;
      }
      public double DistanceBetween2Points(ref Point pt1, ref Point pt2)
      {
        double sqr1 = Math.Pow(pt2.x - pt1.x, 2);
        double sqr2 = Math.Pow(pt2.y - pt1.y, 2);
        double distance = Math.Sqrt(sqr1 + sqr2);
        return distance;
      }
    }
    public class Program
    {
      private static CoordinateSystem IdentifyCoordinates()
      {
        CoordinateSystem coord = new CoordinateSystem();
        Console.WriteLine("Start Point");
        coord.start = coord.GetThePoint();
        Console.WriteLine("End Point");
        coord.end = coord.GetThePoint();
        return coord;
      }
      private static void Show(CoordinateSystem c)
      {
        Console.WriteLine("Coordinate System");
        Console.WriteLine("Starting Point: P({0}, {1})", c.start.x, c.start.y);
        Console.WriteLine("Ending Point:  Q({0}, {1})", c.end.x, c.end.y);
        Console.WriteLine("Distance Between Both Points: {0:F}", c.DistanceBetween2Points(ref c.start, ref c.end));
      }
      static int Main()
      {
        CoordinateSystem coord = IdentifyCoordinates();
        Console.WriteLine();
        Show(coord);
        return 0;
      }
    }
  }
```

بازگرداندن یک شی یا ارسال آن به عنوان آرگومان

۱. برای بازگرداندن کلاس یا ارسال آن به عنوان آرگومان، فایل OrderProcessing.cs را به صورت زیر اصلاح کنید.

```
using System;
  namespace ElectronicStore1
  {
    public class SaleItem
    {
```

```
double DiscountAmount;
double NetPrice;
int Quantity;
double SaleTotal;
public double GetDiscountRate()
{
  Console.Write("Discount Applied (Enter 0 to 100, 0 if no discount): ");
  double discount = double.Parse(Console.ReadLine());
  return discount;
}
public int GetQuantity()
{
  Console.Write("Enter Quantity: ");
  int q = int.Parse(Console.ReadLine());
  return q;
}
public StoreItem Create()
{
  long itemNumber;
  char category;
  string make;
  string model;
  //double discount;
  double price;
  StoreItem saleItem = new StoreItem();
  Console.Write("Enter the Item #: ");
  itemNumber = long.Parse(Console.ReadLine());
  Console.WriteLine("Category");
  Console.WriteLine("A - Audio Cables");
  Console.WriteLine("B - Instructional and Tutorials (Books)");
  Console.WriteLine("C - Cell Phones and Accessories");
  Console.WriteLine("D - Bags and Cases");
  Console.WriteLine("E - Headphones");
  Console.WriteLine("F - Instructional and Tutorials (VHS & DVD)");
  Console.WriteLine("G - Digital Cameras");
  Console.WriteLine("H - Cables and Connectors");
  Console.WriteLine("I - PDAs and Accessories");
  Console.WriteLine("J - Telephones and Accessories");
  Console.WriteLine("K - Surge Protector");
  Console.WriteLine("L - TVs and Videos");
  Console.WriteLine("U - Unknown");
  Console.Write("Your Choice? ");
  category = char.Parse(Console.ReadLine());
  Console.Write("Make:      ");
  make = Console.ReadLine();
  Console.Write("Model:     ");
  model = Console.ReadLine();
  Console.Write("Unit Price: ");
  price = double.Parse(Console.ReadLine());
  saleItem.SetItemNumber(itemNumber);
```

```
        saleItem.SetCategory(category);
        saleItem.SetMake(make);
        saleItem.SetModel(model);
        saleItem.SetUnitPrice(price);
        return saleItem;
    }
    public void ShowSaleItem(StoreItem item)
    {
        double discountRate = GetDiscountRate();
        int quantity = GetQuantity();
        DiscountAmount = item.GetUnitPrice() * discountRate / 100;
        NetPrice = item.GetUnitPrice() - DiscountAmount;
        SaleTotal = NetPrice * quantity;
        Console.WriteLine("\nStore Item Description");
        Console.WriteLine("Item Number:   {0}", item.GetItemNumber());
        Console.WriteLine("Category:     {0}", item.GetCategory());
        Console.WriteLine("Make         {0}", item.GetMake());
        Console.WriteLine("Model:       {0}", item.GetModel());
        Console.WriteLine("Unit Price:   {0:C}", item.GetUnitPrice());
        Console.WriteLine("Discount Rate:  {0:P}", discountRate / 100);
        Console.WriteLine("Discount Amount: {0:C}", DiscountAmount);
        Console.WriteLine("Price/Item:   {0:C}", NetPrice);
        Console.WriteLine("Quantity:     {0}", quantity);
        Console.WriteLine("Sale Total:   {0:C}", SaleTotal);
    }
  }
  public class OrderProcessing
  {
    public static int Main()
    {
      StoreItem item = new StoreItem();
      SaleItem sale = new SaleItem();
      string strTitle1 = "=-= Nearson Electonics =-=\n";
      string strTitle2 = "******* Store Items *******";
      Console.Title = "Electronic Super Store";
      Console.WriteLine(strTitle1);
      Console.WriteLine(strTitle2);
      Console.Clear();
      item = sale.Create();
      sale.ShowSaleItem(item);
      System.Console.ReadKey();
      return 0;
    }
  }
}
```

۲. برنامه را اجرا کنید.

۳. کلید Enter را بزنید.

```
=-= Nearson Electonics =-=
******* Store Items ******
Enter the Item #: 927374
Category
A - Audio Cables
B - Instructional and Tutorials (Books)
C - Cell Phones and Accessories
D - Bags and Cases
E - Headphones
F - Instructional and Tutorials (VHS & DVD)
G - Digital Cameras
H - Cables and Connectors
I - PDAs and Accessories
J - Telephones and Accessories
K - Surge Protector
L - TVs and Videos
U - Unknown
Your Choice? L
Make:      NEC
Model:     VT48 Video Projector
Unit Price: 705.95
Discount Applied (Enter 0 to 100, 0 if no discount): 15
Enter Quantity: 1
```

۴. کلید Enter را فشار دهید.

```
Store Item Description
Category:     L
Make:         NEC
Model:        VT48 Video Projector
Unit Price:   $705.95
Discount Rate:  15.00 %
Discount Amount: $105.89
Price/Item:   $600.06
Quantity:     1
Sale Total:   $600.06
```

۵. با زدن Enter از برنامه خارج شده و به محیط برنامه نویسی بازگردید.

ارسال یک کلاس به عنوان آرگومان خود آن کلاس

می توان نمونه ای از یک کلاس را به عنوان آرگومان به یکی از متدهای خود آن کلاس فرستاد. برای این منظور، آرگومان را مثل هر کلاس دیگری ارسال می کنید.

مثال

```
public class Point
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
{
  internal int x;
  internal int y;
  public void Equivalent(Point Same)
  {
  }
}
```

حال، می توانید هر کاری که می خواهید در بدنه ی متد انجام دهید. همچنین، می توانید آرگومان را مورد استفاده قرار دهید. حال در صورت استفاده از آرگومان گفته شده، لازم است به خاطر داشته باشید، همگی (دیگر) اعضای کلاس از طریق آرگومان قابل دسترسی هستند. ساده ترین روش استفاده از آرگومان، تخصیص (هر یک از) مقادیر آن به عضو های معادل و مربوط در کلاس است.

مثال

```
public class Point
{
  internal int x;
  internal int y;
  public void Equivalent(Point Same)
  {
    this.x = Same.x;
    this.y = Same.y;
  }
}
```

هنگام فراخوانی متد، لازم است نمونه ای از کلاس به آن ارسال کنید. در وهله ی اول کلاس مورد نظر را ایجاد و تعریف می کنیم، بعد آن را ارسال می کنیم.

مثال

```
using System;
public class Point
{
  internal int x;
  internal int y;
  public void Equivalent(Point Same)
  {
    this.x = Same.x;
    this.y = Same.y;
  }
}
public class Program
{
  private static void ShowPoint(Point pt)
  {
    Console.Write("Point Coordinates: ");
```

```
    Console.WriteLine("A({0}, {1})", pt.x, pt.y);
  }
  static int Main(string[] args)
  {
    Point pt = new Point();
    pt.x = 4;
    pt.y = 6;
    ShowPoint(pt);
    Point One = new Point();
    One.Equivalent(pt);
    ShowPoint(One);
    return 0;
  }
}
```

نتیجه ی زیر حاصل می گردد.

```
Point Coordinates: A(4, 6)
Point Coordinates: A(4, 6)
Press any key to continue...
```

به جای اینکه اول کلاس را تعریف و مقداردهی اولیه کنید، می توانید نمونه ای از کلاس مورد نظر را داخل پرانتز های متد فراخوان (calling method) ایجاد کنید. البته برای این منظور نیاز به سازنده ای (constructor) دارید که مقادیر فیلد های کلاس مربوط را مشخص کرده تا آرگومان به تناسب (با مقدار مناسب) مقداردهی اولیه شود.

مثال

```
using System;
public class Point
{
  internal int x;
  internal int y;
  public Point()
  {
  }
  public Point(int XCoord, int YCoord)
  {
    this.x = XCoord;
    this.y = YCoord;
  }
  public void Equivalent(Point Same)
  {
    this.x = Same.x;
    this.y = Same.y;
  }
```

```
}
public class Program
{
  private static void ShowPoint(Point pt)
  {
    Console.Write("Point Coordinates: ");
    Console.WriteLine("A({0}, {1})", pt.x, pt.y);
  }
  static int Main(string[] args)
  {
    Point pt = new Point();
    pt.x = 4;
    pt.y = 6;
    ShowPoint(pt);
    Point One = new Point();
    One.Equivalent(new Point(-3, 2));
    ShowPoint(One);
    return 0;
  }
}
```

به جای متد رسمی (formal method)، می توان سازنده ای از کلاس را برای ارسال نمونه ای از همان کلاس به کار برد. حال می توانید در سازنده، آرگومان را بر اساس نیازتان به کار ببرید (نظر به اینکه تمامی اعضای کلاس در دسترس هستند).

مثال

```
public class Point
{
  internal int x;
  internal int y;
  public Point()
  {
  }
  public Point(int XCoord, int YCoord)
  {
    this.x = XCoord;
    this.y = YCoord;
  }
  public Point(Point Same)
  {
    this.x = Same.x;
    this.y = Same.y;
  }
}
```

هدف ما از ارسال یک کلاس به متدهای خود همان کلاس پیدا کردن معادل آن نیست. زبان C# در حقیقت .NET Framework، با استفاده از متد توکار Equals() ترتیب فرایند بالا را می دهد. در عوض، می توان متدی ایجاد کرد که نمونه ای از همان کلاس را می گیرد ولی نمونه ی نام برده را اصلاح می کند. برای مثال، شاید بخواهیم برای کلاس Point، point جدیدی ایجاد کنیم که به اندازه ی یک واحد از شی Point موجود فاصله داشته باشد.

```
using System;
public class Point
{
  internal int x;
  internal int y;
  public Point()
  {
  }
  public Point(int XCoord, int YCoord)
  {
    this.x = XCoord;
    this.y = YCoord;
  }
  public void Equivalent(Point Same)
  {
    this.x = Same.x;
    this.y = Same.y;
  }
  public void CreatePointOneUnitAway(Point AddUnit)
  {
    this.x = AddUnit.x + 1;
    this.y = AddUnit.y + 1;
  }
}
public class Program
{
  private static void ShowPoint(Point pt)
  {
    Console.Write("Point Coordinates: ");
    Console.WriteLine("A({0}, {1})", pt.x, pt.y);
  }

  static int Main(string[] args)
  {
    Point pt = new Point();
    pt.x = 4;
    pt.y = 6;
    ShowPoint(pt);
    Point One = new Point();
    One.CreatePointOneUnitAway(pt);
    ShowPoint(One);
```

```
        One.CreatePointOneUnitAway(new Point(-8, -3));
        ShowPoint(One);
        return 0;
    }
}
```

نتیجه ی زیر به دست می آید.

```
Point Coordinates: A(4, 6)
Point Coordinates: A(5, 7)
Point Coordinates: A(-7, -2)
Press any key to continue...
```

بازگرداندن کلاسی از متد همان کلاس

می توانید متدی در کلاس ایجاد کنید که نمونه ای از همان کلاس را بر می گرداند. برای شروع کار، اسم کلاس مربوط را در سمت چپ متد وارد کنید.

مثال

```
public class Point
{
    public Point MethodName()
    {
    }
}
```

می توانید کارهای متفاوتی با متد موردنظر انجام دهید. اگر می خواهید مقدار جدیدی از کلاس باز گردانید، می توانید نمونه ای از کلاس را تعریف و مقداردهی اولیه کرده، سپس آن را برگردانید. به مثال زیر توجه کنید.

```
using System;
public class Point
{
    internal int x;
    internal int y;
    public Point()
    {
    }
    public Point(int XCoord, int YCoord)
    {
        this.x = XCoord;
        this.y = YCoord;
    }
    public Point(Point Same)
    {
        this.x = Same.x;
```

```
    this.x = Same.x;
  }
  public Point AdvanceBy5()
  {
    Point Some = new Point();
    Some.x = 5;
    Some.y = 5;
    return Some;
  }
}
public class Program
{
  private static void ShowPoint(Point pt)
  {
    Console.Write("Point Coordinates: ");
    Console.WriteLine("A({0}, {1})", pt.x, pt.y);
  }
  static int Main(string[] args)
  {
    Point pt = new Point();
    pt.x = 4;
    pt.y = 6;
    ShowPoint(pt);
    Point Away5 = pt.AdvanceBy5();
    ShowPoint(Away5);
    return 0;
  }
}
```


نتیجه

```
Point Coordinates: A(4, 6)
Point Coordinates: A(5, 5)
Press any key to continue...
```

می توانید نمونه ای از کلاس را تعریف کرده، مقادیر جاری کلاس را همراه با مقدارهای نمونه ی کلاس به کار ببرید تا مقادیر جدیدی به دست آورید، سپس نمونه ی مذکور را برگردانید.

مثال

```
using System;
public class Point
{
  internal int x;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    internal int y;
    public Point()
    {
    }
    public Point(int XCoord, int YCoord)
    {
      this.x = XCoord;
      this.y = YCoord;
    }
    public Point(Point Same)
    {
      this.x = Same.x;
      this.x = Same.x;
    }
    public Point AdvanceBy5()
    {
      Point Some = new Point();
      Some.x = this.x + 5;
      Some.y = this.y + 5;
      return Some;
    }
  }
  public class Program
  {
    private static void ShowPoint(Point pt)
    {
      Console.Write("Point Coordinates: ");
      Console.WriteLine("A({0}, {1})", pt.x, pt.y);
    }
    static int Main(string[] args)
    {
      Point pt = new Point();
      pt.x = 4;
      pt.y = 6;
      ShowPoint(pt);
      Point Away5 = pt.AdvanceBy5();
      ShowPoint(Away5);
      return 0;
    }
  }
```

نتیجه ی زیر را به دست می دهد.

```
Point Coordinates: A(4, 6)
Point Coordinates: A(9, 11)
Press any key to continue...
```

به یاد داشته باشید که برای فراخوانی متد، چنانچه متد ایستا نبود، لازم است نمونه ی کلاس را از آن جایی که متد را فرا می خوانید، تعریف کنید. نوع دیگر پیاده سازی (روش دوم آن) اصلاح نمونه ای است که متد را فرا می خواند. برای مثال، می توانید مقادیری به فیلدهای آن اضافه کنید، یا هر عملیات دیگری که نیاز دارید روی اعضای نمونه ی فراخوان (calling instance) پیاده کنید.

مثال

```
using System;
public class Point
{
  internal int x;
  internal int y;
  public Point()
  {
  }
  public Point(int XCoord, int YCoord)
  {
    this.x = XCoord;
    this.y = YCoord;
  }
  public Point(Point Same)
  {
    this.x = Same.x;
    this.x = Same.x;
  }
  // This method adds 1 to each field of the class
  // to get a new point away North-East of the current point
  public Point CreatePointOneUnitAway()
  {
    this.x = this.x + 1;
    this.y = this.y + 1;
    return this;
  }
}
public class Program
{
  private static void ShowPoint(Point pt)
  {
    Console.Write("Point Coordinates: ");
    Console.WriteLine("A({0}, {1})", pt.x, pt.y);
  }
  static int Main(string[] args)
  {
    Point pt = new Point();
    pt.x = 4;
    pt.y = 6;
    ShowPoint(pt);
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
        Point One = new Point(-8, 5);
        Point Another = One.CreatePointOneUnitAway();
        ShowPoint(Another);
        return 0;
    }
}
```

نتیجه ی زیر حاصل می گردد.

```
Point Coordinates: A(4, 6)
Point Coordinates: A(-7, 6)
Press any key to continue...
```

شایان توجه است که می توان متدی ساخت که علاوه بر گرفتن آرگومان با کلاس والد خود هم نوع باشد. در متد، می توان به تمامی اعضای کلاس دسترسی داشت که از جمله ی آن فراخواندن دیگر متدهای کلاس مربوط می باشد.

مقدمه ای بر شرطی ها

متغیر های Boolean

هنگام تعامل با کامپیوتر، کاربر ممکن است مقادیری را به برنامه ی در حال اجرا عرضه کند که برخی از این مقادیر معتبر هستند ولی برخی دیگر نیاز به اصلاحاتی دارند یا توسط رایانه رد می شوند. برای جلوگیری از این رخداد مقادیر باید مرتب مورد بررسی قرار بگیرند. اعتبار یک مقدار در رابطه با نوعش بررسی می شود. برای مثال، یک شرط می تواند درست باشد یا یک عدد مساوی عدد دیگری باشد.

برای اعتبارسنجی مقادیر، زبان C# یک سری نشانه عرضه می کند که از آن ها به عنوان عملگرهای Boolean یاد می شود.

تعریف متغیرهای Boolean

۱. Microsoft Visual Studio را اجرا کنید.

۲. برای ایجاد برنامه ی جدید، به فهرست اصلی مراجعه کرده، روی File -> New Project... کلیک کنید.

۳. در فهرست میانی روی گزینه ی Empty Project کلیک کنید.

۴. اسم مورد نظررا به NationalBank2 تغییر دهید.

۵. برای ایجاد فایل جدید، در پنجره ی Solution Explorer، راست کلیک کرده سپس NationalBank2 -> Add -> New Item...

۶. گزینه ی Code File را از لیست میانی انتخاب کنید.

۷. اسم فایل را به Employee تغییر دهید.

۸. Add را کلیک کنید.

۹. فایل Employee.cs را به شکل زیر اصلاح کنید.

```
public class Employee
{
  public string EmployeeNumber;
  public string FirstName;
  public string LastName;
  public double HourlySalary;
  public Employee(string emplNbr = "00-000", string fName = "John",
          string lName = "Doe", double salary = 0.00)
  {
    EmployeeNumber = emplNbr;
    FirstName = fName;
    LastName = lName;
    HourlySalary = salary;
  }
}
```

۱۰. برای ایجاد فایل جدید، در پنجره ی Solution Explorer، راست کلیک کرده سپس : NationalBank2 -> Add -> New Item...

۱۱. توجه داشته باشید که در لیست میانی گزینه ی Code File انتخاب شده باشد. اسم آن را به Management تغییر دهید و کلید Enter را بزنید.

۱۲. فایل را به صورت زیر اصلاح کنید.

```
using System;
public class Management
{
  private static Employee HireEmployee()
  {
    Employee empl = new Employee();
    Console.Title = "National Bank";
    Console.WriteLine("========================");
    Console.WriteLine("==-= National Bank =-==");
    Console.WriteLine("To hire a new employee, enter the following information");
    Console.WriteLine("------------------------");
    Console.Write("Employee #:  ");
    empl.EmployeeNumber = Console.ReadLine();
    Console.Write("First Name:  ");
    empl.FirstName = Console.ReadLine();
    Console.Write("Last Name:  ");
    empl.LastName = Console.ReadLine();
    Console.Write("Hourly Salary: ");
```

```
    empl.HourlySalary = double.Parse(Console.ReadLine());
    Console.WriteLine("========================");
    return empl;
  }
  private static void ShowEmployeeRecord(Employee empl)
  {
    Console.Title = "National Bank";
    Console.WriteLine("========================");
    Console.WriteLine("==-= National Bank =-==");
    Console.WriteLine("   Employee Record");
    Console.WriteLine("------------------------");
    Console.WriteLine("Employee #:  {0}", empl.EmployeeNumber);
    Console.WriteLine("First Name:  {0}", empl.FirstName);
    Console.WriteLine("Last Name:   {0}", empl.LastName);
    Console.WriteLine("Hourly Salary: {0}", empl.HourlySalary);
    Console.WriteLine("========================");
  }
  public static int Main()
  {
    Employee clerk = null;
    clerk = HireEmployee();
    Console.Clear();
    ShowEmployeeRecord(clerk);
    Console.ReadKey();
    return 0;
  }
}
```

۱۳. برای اجرای برنامه، به فهرست اصلی مراجعه کرده، سپس روی Debug -> Start Debugging کلیک کنید.

۱۴. اطلاعات لازم را به صورت زیر اصلاح کنید.

Employee #	88-602
First Name	Joan
Last Name	Gergman
Hourly Salary	16.85

۱۵. کلید Enter را فشار دهید.

```
======================
==-= National Bank =-==
Employee Record
----------------------
Employee #:  88-602
First Name:  Joan
 Last Name:  Bergman
 Hourly Salary: 16.85
 ======================
```

۱۶. حال Enter را زده تا پنجره ی DOS بسته شود.

تعریف متغیر Boolean

به متغیری Boolean گفته می شود که مقدار آن از دو حالت true یا false خارج نباشد. برای تعریف متغیر Boolean، می توان هر یک از دو کلیدواژه ی var یا bool را به کار برد. به مثال زیر توجه کنید.

```
using System;
public class Exercise
{
  static int Main()
  {
    bool drinkingUnderAge;
    return 0;
  }
}
```

می توان از نوع داده ی Boolean برای تعریف متغیر Boolean استفاده کرد. نوع داده ی Boolean بخشی از فضای نام System می باشد.

مثال

```
using System;
public class Exercise
{
  static int Main()
  {
    bool drinkingUnderAge;
    Boolean theFloorIsCoveredWithCarpet;
    return 0;
  }
}
```

پس از تعریف متغیر، باید آن را با (مقدار) true یا false مقداردهی اولیه کرد. در حقیقت، چنانچه متغیر نام برده را به عنوان var تعریف کنید، چاره ای جز مقداردهی اولیه ی آن ندارید. توجه خود را به مثال زیر جلب کنید.

```
using System;
public class Exercise
{
  static int Main()
  {
    var drinkingUnderAge = true;
    return 0;
  }
}
```

به منظور نمایش مقدار متغیر Boolean در کنسول، می توانید اسم آن را در پرانتزهای متد Write() یا WriteLine() کلاس console تایپ کنید. مثال زیر را در نظر بگیرید.

```
using System;
public class Exercise
{
  static int Main()
  {
    var drinkingUnderAge = true;
    Console.WriteLine("Drinking Under Age: {0}", drinkingUnderAge);
    return 0;
  }
}
```

نتیجه

```
Drinking Under Age: True
Press any key to continue...
```

می توانید هر زمان که لازم می دانید، مقدار متغیر Boolean را با تخصیص مقادیر true یا false تغییر دهید. مثال

```
using System;
public class Exercise
{
  static int Main()
  {
    var drinkingUnderAge = true;
    Console.WriteLine("Drinking Under Age: {0}", drinkingUnderAge);
    drinkingUnderAge = false;
    Console.WriteLine("Drinking Under Age: {0}", drinkingUnderAge);
    return 0;
  }
}
```

نتیجه ی زیر حاصل می گردد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
Drinking Under Age: True
Drinking Under Age: False
Press any key to continue...
```

بازیابی مقدار متغیر Boolean

می توان مقدار یک متغیر Boolean را از کاربر در خواست کرد. در این مورد، لازم است کاربر True (یا true) یا False (یا false) را تایپ کند. برای بازیابی آن باید از متدهای Read() یا ReadLine() کلاس Console کمک گرفت.

مثال

```
using System;
public class Exercise
{
  static int Main()
  {
    var drivingUnderAge = false;
    Console.WriteLine("Were you driving under age?");
    Console.Write("If Yes, enter True. Otherwise enter False: ");
    drivingUnderAge = bool.Parse(Console.ReadLine());
    Console.WriteLine("\nWas Driving Under Age: {0}\n", drivingUnderAge);
    return 0;
  }
}
```

مثال

```
Were you driving under age?
If Yes, enter True. Otherwise enter False: true
 as Driving Under Age: True
 ress any key to continue...
```

ایجاد فیلد Boolean

مثل دیگر نوع متغیر هایی که در مباحث پیشین با آن برخورد داشتیم، می توان متغیر Boolean را به فیلد یک کلاس تبدیل کرد. متغیر مذکور را مثل هر نوع متغیر دیگر، البته با کلید واژه ی bool یا نوع داده ی Boolean تعریف می کنید.

مثال

```
public class House
{
  char typeOfHome;
```

```
  int beds;
  float baths;
  byte stories;
  bool hasCarGarage;
  int yearBuilt;
  double value;
}
```

هنگام مقداردهی اولیه ی شی ای که متغیر Boolean یک عضو آن محسوب می شود، کافی است true یا false به متغیر مربوط اختصاص دهید. به همین نحو، می توانید مقدار متغیر عضو Boolean را با دسترسی به آن بازیابی یا چک کنید. نمونه های آن را در زیر مشاهده می کنید.

```
using System;
public class House
{
  public char typeOfHome;
  public int beds;
  public float baths;
  public byte stories;
  public bool hasCarGarage;
  public int yearBuilt;
  public double value;
}
public class Program
{
  static int Main()
  {
    var Condominium = new
    {
      hasCarGarage = false,
      yearBuilt = 2002,
      baths = 1.5F,
      stories = 18,
      value = 155825,
      beds = 2,
      typeOfHome = 'C'
    };
    Console.WriteLine("=//= Altair Realtors =//=");
    Console.WriteLine("=== Property Listing ===");
    Console.WriteLine("Type of Home:     {0}", Condominium.typeOfHome);
    Console.WriteLine("Number of Bedrooms: {0}", Condominium.beds);
    Console.WriteLine("Number of Bathrooms: {0}", Condominium.baths);
    Console.WriteLine("Number of Stories:  {0}", Condominium.stories);
    Console.WriteLine("Year Built:       {0}", Condominium.yearBuilt);
    Console.WriteLine("Has Car Garage:    {0}", Condominium.hasCarGarage);
    Console.WriteLine("Monetary Value:    {0}\n", Condominium.value);
    return 0;
  }
```

```
}
```

نتیجه ی زیر به دست می آید.

```
=//= Altair Realtors =//=
=== Property Listing ===
Type of Home:      C
Number of Bedrooms: 2
Number of Bathrooms: 1.5
Number of Stories:  18
Year Built:        2002
Has Car Garage:    False
Monetary Value:    155825
Press any key to continue...
```

آرگومان های Boolean

درست مثل پارامترهای انواع دیگر، می توان آرگومان (از) نوع bool یا Boolean به متد ارسال کرد. چنین آرگومانی می تواند تنها دو مقدار true یا false را داشته باشد.

enumeration

تصور کنید، هنگام برنامه نویسی برای شرکتی که معاملات املاک انجام می دهد، می خواهید برنامه ای که نوشته اید از مشتری نوع خانه ای را که قصد خرید آن را دارند بپرسد.

مثال

```
using System;
public class Exercise
{
  static int Main()
  {
    int typeOfHouse = 0;
    int typeOfGarage = 0;
    Console.WriteLine("Enter the type of house you want to purchase");
    Console.WriteLine("1 - Single Family");
    Console.WriteLine("2 - Townhouse");
    Console.WriteLine("3 - Condominium");
    Console.Write("Your Choice: ");
    typeOfHouse = int.Parse(Console.ReadLine());
    Console.WriteLine("Enter the type of garage you want");
    Console.WriteLine("0 - Doesn't matter");
```

```
        Console.WriteLine("1 - Interior");
        Console.WriteLine("2 - Exterior");
        Console.Write("Your Choice: ");
        typeOfGarage = int.Parse(Console.ReadLine());
        Console.Clear();
        Console.WriteLine("\nHouse Type: {0}", typeOfHouse);
        Console.WriteLine("Garage Type: {0}", typeOfGarage);
        return 0;
    }
}
```

مثالی از اجرای برنامه

```
Enter the type of house you want to purchase
 1 - Single Family
 2 - Townhouse
 3 - Condominium
 Your Choice: 3
Enter the type of garage you want
0 - Doesn't matter
1 - Interior
2 - Exterior
Your Choice: 1
--------------------------------------------------------
House Type: 3
Garage Type: 1
Press any key to continue...
```


برای چنین برنامه ای، ارقام ممکن است گنگ یا مبهم باشد. عدد ۱، رقمی کلی است ولی در برنامه ی مورد نظر ممکن است نشانگر خانه ای با ظرفیت گنجایش تنها یک خانوار باشد. حال برنامه ی نام برده از ثابت ۱ در راه های معنی دار مختلفی بهره می برد. برای ایجاد امکان دادن بیش از یک معنی یا کاربرد به یک عدد ثابت، به خصوص هنگامی که قرار است عدد مربوطه جزئی از یک سری باشد، زبان برنامه نویسی C# به شما اجازه می دهد یک نوع لیست ایجاد کنید.

enumeration در واقع یک سری عدد صحیح ثابت (constant integer) است که هر یک موقعیت خاصی در لیست دارد و با اسم مشخصی شناسایی می شود. بر این اساس، به جای به خاطر داشتن این که ثابت ۱ نشانگر تک خانوار است، می توان لیستی ایجاد کرد که نمونه ی خانه ی مزبور را شامل می شود. در لیست دیگر، به جای استفاده ی دوباره از ۱، می توان اسم برای آن انتخاب کرد. در نتیجه، در هر لیست ، اگرچه ثابت ۱ باز هم در نظر گرفته می شود، حداقل معنای آن دقیق خواهد بود.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

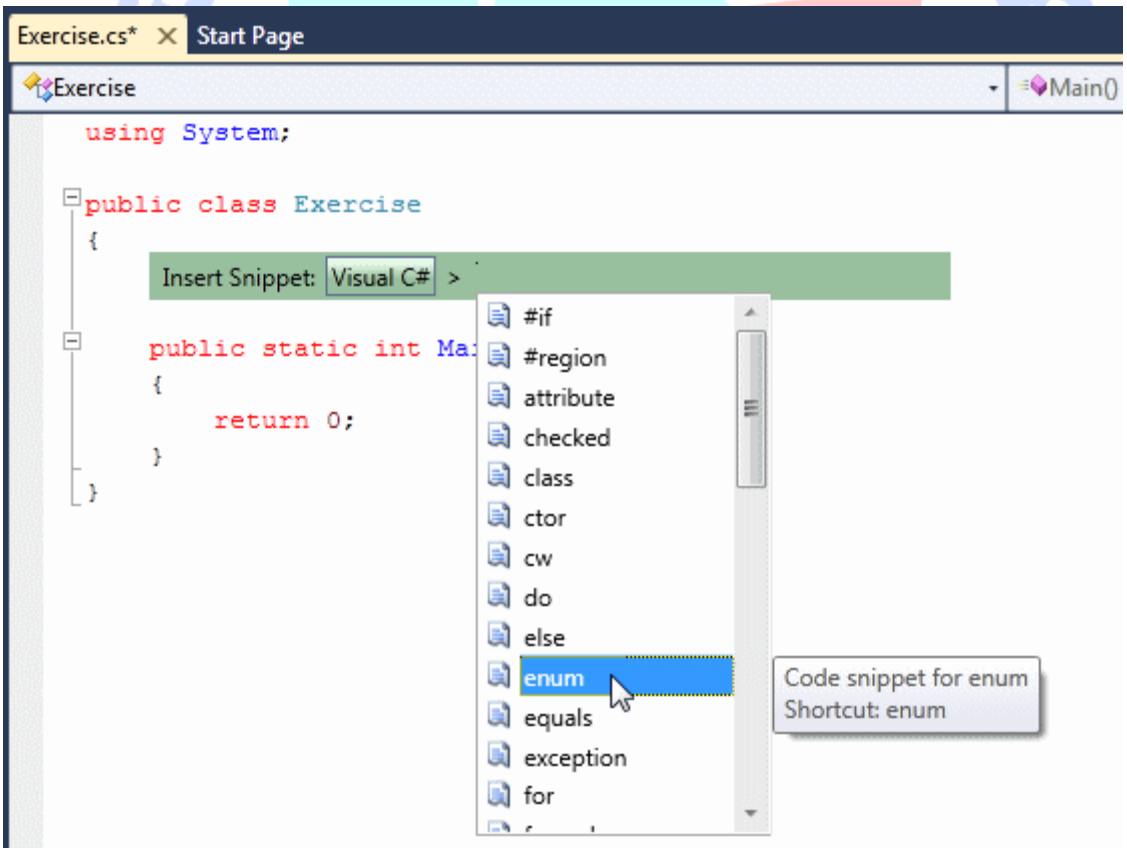
ایجاد enumeration

برای این منظور، کلیدواژه ی enum و به دنبال آن اسم enumeration و همچنین اسمی مختص هر یک از آیتم های لیست را تایپ کنید. اسم شمارنده (enumerator) و همچنین اسم تک تک آیتم های لیست از همان قوانینی که برای انتخاب اسم (ها) نام بردیم پیروی می کند.

فرمول ایجاد enumeration به شرح زیر است.

```
enum Enumeration_Name {Item1، Item2، Item_n};
```

برای ایجاد enumeration، می توانید کد را به صورت دستی وارد کنید. برای استفاده از code snippet، روی بخش مورد نظر راست کلیک کرده، سپس ...Insert Snippet را انتخاب کنید. اکنون روی Visual C# دوبار کلیک کنید. در لیستی که نمایان می شود، روی enum دوبار کلیک کنید.



مثال

```
using System;
public class Exercise
{
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
  enum HouseType { Unknown, SingleFamily, TownHouse, Condominium }
  static int Main()
  {
    return 0;
  }
}
```

تعریف متغیر enumeration

پس از ایجاد enumeration، هر یک از اعضای آن، مقدار عدد طبیعی مثل 0، ۴، 12، 25 خواهند داشت.

پس از ایجاد enumeration، حتی می توان از آن متغیر تعریف کرد. به مثال زیر توجه کنید.

```
using System;
public class Exercise
{
  enum HouseType { Unknown, SingleFamily, TownHouse, Condominium }
  static int Main()
  {
    HouseType propType;
    return 0;
  }
}
```

لازم به ذکر است که می توان متغیر (از) نوع enumeration را با کلید واژه ی var نیز تعریف کرد.

مقداردهی اولیه ی متغیر enumeration

پس از تعریف متغیر برای نوع enumeration، به منظور مقداردهی اولیه ی آن، مشخص کنید کدام عضو enumeration به متغیر اختصاص داده شود. شایان توجه است که تنها عضو شناس enumeration باید به متغیر تخصیص یابد. برای این منظور، در سمت راست عملگر جایگزین، اسم enumeration و به دنبال آن عملگر نقطه، عضوی که می خواهید مقدار آن را اختصاص دهید، تایپ کنید.

مثال

```
using System;
public class Exercise
{
  enum HouseType { Unknown, SingleFamily, TownHouse, Condominium }
  static int Main()
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    {
      var propType = HouseType.SingleFamily;
      return 0;
    }
}
```

همچنین می توان مقداری را که متغیر تعریف شده در حال حاضر دارد، به دست آورد. برای مثال، می توان آن را در کنسول و با استفاده از متدهای Write() و WriteLine() نمایش داد. به مثال زیر توجه کنید.

```
using System;
public class Exercise
{
  enum HouseType { Unknown, SingleFamily, TownHouse, Condominium }
  static int Main()
  {
    var propType = HouseType.SingleFamily;
    Console.WriteLine("House Type: {0}", propType);
    return 0;
  }
}
```

نتیجه ی زیر را به دست می دهد.

```
House Type: SingleFamily
Press any key to continue...
```

enumeration در واقع لیستی از اعداد است که هر یک توسط اسم معینی شناسایی می شود. به صورت پیش فرض، اولین آیتم لیست مقدار ۰ را دارد، دومی مقدار ۱ و غیره... برای مثال، در HouseType enumeration، Unknown مقدار ۰ را دارد، در حالی که عضو Townhouse مقدار ۲. مقادیر گفته شده پیش فرض هستند. چنانچه، با مقادیر بالا موافق نیستید می توانید مقدار هر یک از اعضای لیست را خود تعیین کنید. برای مثال، اگر بخواهیم عضو Unknown در enumeration فوق مقدار ۵ داشته باشد، کافی است عملگر جایگزین " = " را به کار ببرید. شمارنده به این ترتیب خواهد بود.

```
using System;
public class Exercise
{
  enum HouseType { Unknown = 5, SingleFamily, TownHouse, Condominium }
  static int Main()
  {
    return 0;
  }
}
```

در این مورد عضو Unknown، مقدار ۵ را خواهد داشت، SingleFamily مقدار ۶، به این خاطر که پس از عضوی قرار گرفته که مقدار آن ۱ می باشد (بنابراین 6 = 1 + 5). عضو Townhouse مقدار ۷ و Condominium مقدار ۸ را خواهد داشت. همچنین می توان مقداری را به چندین عضو enumeration اختصاص داد.

مثال

```
using System;
public class Exercise
{
  enum HouseType { Unknown = 3, SingleFamily = 12, TownHouse, Condominium = 8 }
  static int Main()
  {
    return 0;
  }
}
```

در این مثال، Townhouse مقدار ۱۳ را خواهد داشت زیرا پس از عضو SingleFamily قرار گرفته که مقدار آن ۱۲ می باشد.

قابلیت رویت، دسترسی به enumeration

به صورت پیش فرض، پس از ایجاد enumeration، دسترسی به آن تنها از طریق پروژه ای امکان پذیر می باشد که در آن، enumeration ایجاد شده است. درست مثل کلاس، می توان (سطح دسترسی) دسترسی به enumeration را بیرون از پروژه اش مدیریت و تنظیم کرد. به عبارت دیگر، می توان enumeration را پنهان کرد یا برعکس بیرون از پروژه اش آن را قابل رویت ساخت. برای این منظور، می توانید کلیدواژه ی های private و public را پیش از enumeration قرار دهید. نظر خود را به مثال زیر جلب کنید.

```
using System;
public class Exercise
{
  public enum HouseType
  {
    Unknown,
    SingleFamily,
    TownHouse,
    Condominium
  }
  static int Main()
  {
    HouseType propType = HouseType.SingleFamily;
    Console.WriteLine("House Type: {0}", propType);
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    return 0;
  }
}
```

enumeration به عنوان متغیر عضو

پس از ایجاد enumeration، می توان از آن به عنوان یک نوع داده برای تعریف متغیر استفاده کرد. برای ایجاد فیلد (از) نوع enumeration، باید همان قوانینی را که برای داده های نوع اولیه رعایت می کردیم پیاده کنیم : اسم enumeration، به دنبال آن اسم متغیر و نقطه ویرگول " ; ". به مثال زیر توجه کنید :

```
public enum HouseType
{
  Unknown,
  SingleFamily,
  TownHouse,
  Condominium
}
public class House
{
  HouseType propertyType;
}
```

به همین ترتیب، می توان هر تعداد متغیر عضو که لازم است ایجاد کرد. پس از تعریف متغیر، برای مقداردهی اولیه ی آن، عضو دلخواه enumeration (enumeration member) را به متغیر اختصاص دهید.

مثال

```
public enum HouseType
{
  Unknown,
  SingleFamily,
  TownHouse,
  Condominium
}
public class House
{
  HouseType propertyType;
  public House()
  {
    propertyType = HouseType.Unknown;
  }
}
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

پس از مقداردهی اولیه متغیر عضو، می توانید آن را هر طور که مایلید مورد استفاده قرار دهید. برای مثال، به منظور نمایش مقدار متغیر عضو می توان آن را به متدهای Write() یا Writeline() ارسال کرد. به مثال زیر توجه کنید.

```
using System;
public enum HouseType
{
  Unknown,
  SingleFamily,
  TownHouse,
  Condominium
}
public class House
{
  public HouseType propertyType;
  public House()
  {
    propertyType = HouseType.Unknown;
  }
  public void Display()
  {
    Console.WriteLine("Property Type: {0}", propertyType);
  }
}
public class Exercise
{
  static int Main()
  {
    var propType = new House();
    propType.Display();
    Console.WriteLine();
    propType.propertyType = HouseType.SingleFamily;
    propType.Display();
    Console.WriteLine();
    return 0;
  }
}
```

نتیجه ی زیر حاصل می شود.

```
Property Type: Unknown
Property Type: SingleFamily
Press any key to continue...
```

چنانچه از آن به عنوان یک نوع داده ی ساده استفاده کنید، می توانید متدی ایجاد کنید که enumeration بازمی گرداند. همچنین می توانید enumeration را به عنوان یک آرگومان به متد ارسال کنید.

ارسال enumeration به عنوان آرگومان

همان طور که پیش تر ذکر شد، پس از این که enumeration ایجاد می شود، دیگر یک نوع تلقی می گردد. می توان آن را به عنوان آرگومان ارسال کرد یا آن را از یک متد بازگرداند.

Enumeration به همان شکلی که یک نوع داده ی معمولی فرستاده می شود، باید به عنوان آرگومان ارسال گردد.

مثال

```
public class Exercise
{
  private static void ShowHouse(HouseType propType)
  {
  }
}
```

به همین ترتیب، می توان هر تعداد نوع شمارشی که لازم است ارسال کرد. می توان enumeration را در بدنه ی رویه (procedure) به کاربرد و مورد استفاده قرار داد یا آن را نادیده گرفت. هنگام فراخوانی رویه، آرگومانی را ارسال کنید که نوع مقدارش از نوع شمارشی باشد.

مثال

```
using System;
public enum HouseType
{
  Unknown = 2,
  SingleFamily = 4,
  TownHouse = 6,
  Condominium = 8,
}
public class Exercise
{
  private static void ShowHouse(HouseType propType)
  {
    Console.WriteLine("Type of house: {0}", propType);
  }
  public static int Main()
  {
    HouseType ht;
    ht = HouseType.SingleFamily;
    ShowHouse(ht);
    Console.ReadKey();
    return 0;
  }
}
```

نتیجه

```
Type of house: SingleFamily
Press any key to continue...
```

ارسال آرگومان می تواند اختیاری باشد. هنگام ایجاد متد، برای این که نشان دهیم آرگومان مقدار پیش فرض دارد و عضو دلخواه را به آن اختصاص دهیم از عملگر جایگزین " = " استفاده می کنیم. هنگام فراخوانی متد، می توان مقدار برای آرگومان گفته شده ارسال کرد یا از ارسال آن صرف نظر کرد.

مثال

```
using System;
public enum HouseType
{
   Unknown = 2,
   SingleFamily = 4,
   TownHouse = 6,
   Condominium = 8,
}
public class Exercise
{
   private static void ShowHouse(HouseType propType = HouseType.Unknown)
   {
      Console.WriteLine("Type of house: {0}", propType);
   }
   public static int Main()
   {
      HouseType ht;
      ht = HouseType.SingleFamily;
      ShowHouse();
      return 0;
   }
}
```

نتیجه ی زیر به دست می آید.

```
Type of house: Unknown
Press any key to continue...
```

تعریف و استفاده از نوع داده ای enum

۱. فایل Employee.cs را باز کنید.

۲. برای ایجاد تعدادی enumeration، فایل مذکور را به صورت زیر ویرایش نمایید:

```
public enum EmploymentStatus
 {
   FullTime,
   PartTime,
   Internship,
   Unknown
 }
 public class Employee
 {
   public string EmployeeNumber;
   public string FirstName;
   public string LastName;
   public string Title;
   public double HourlySalary;
   public EmploymentStatus Status;
   public Employee(string emplNbr = "00-000", string fName = "John",
            string lName = "Doe", string function = "Temporary",
            double salary = 0.00,
            EmploymentStatus emplStatus = EmploymentStatus.Unknown)
   {
     EmployeeNumber = emplNbr;
     FirstName = fName;
     LastName = lName;
     Title = function;
     HourlySalary = salary;
     Status = emplStatus;
   }
 }
```

۳. حال فایل Management.cs را باز نمایید.

۴. برای استفاده از enumeration، فایل را به صورت زیر ویرایش کنید:

```
using System;
 public class Management
 {
   private static Employee HireEmployee()
   {
     int? status = null;
     Employee empl = new Employee();

     Console.Title = "National Bank";
     Console.WriteLine("========================================");
     Console.WriteLine("==-= National Bank =-==");
     Console.WriteLine("To hire a new employee, enter the following information");
     Console.WriteLine("----------------------------------------");
     Console.Write("Employee #:  ");
     empl.EmployeeNumber = Console.ReadLine();
```

```
    Console.Write("First Name:  ");
    empl.FirstName = Console.ReadLine();
    Console.Write("Last Name:  ");
    empl.LastName = Console.ReadLine();
    Console.Write("Title:  ");
    empl.Title = Console.ReadLine();
    Console.Write("Hourly Salary: ");
    empl.HourlySalary = double.Parse(Console.ReadLine());
    Console.WriteLine("Employment Status");
    Console.WriteLine("1. Full-Time");
    Console.WriteLine("2. Part-Time");
    Console.WriteLine("3. Intern");
    Console.Write("Your Choice? ");
    Status = int.Parse(Console.ReadLine());
    empl.status = (EmploymentStatus)(status - 1);
    Console.WriteLine("========================================");
    return empl;
  }

  private static void ShowEmployeeRecord(Employee empl)
  {
    Console.Title = "National Bank";
    Console.WriteLine("========================================");
    Console.WriteLine("==-= National Bank =-==");
    Console.WriteLine("   Employee Record");
    Console.WriteLine("----------------------------------------");
    Console.WriteLine("Employee #:  {0}", empl.EmployeeNumber);
    Console.WriteLine("First Name:  {0}", empl.FirstName);
    Console.WriteLine("Last Name:  {0}", empl.LastName);
    Console.WriteLine("Employee Title:  {0}", empl.Title);
    Console.WriteLine("Hourly Salary:  {0}", empl.HourlySalary);
    Console.WriteLine("Employment Status: {0}", empl.Status);
    Console.WriteLine("========================================");
  }
  public static int Main()
  {
    Employee clerk = null;
    clerk = HireEmployee();
    Console.Clear();
    ShowEmployeeRecord(clerk);
    Console.ReadKey();
    return 0;
  }
}
```

۵. برای اجرای برنامه، کلید F5 را فشار دهید.

۶. اطلاعات زیر را وارد نمایید:

Employee #	42-855
First Name	Paula
Last Name	Meyer
Title	Cashier
Hourly Salary	18.25
Employment Status	2

```
========================================
==-= National Bank =-==
To hire a new employee, enter the following information
----------------------------------------
Employee #:  42-855
First Name:  Paula
Last Name:  Meyer
Title:  Cashier
Hourly Salary: 18.25
Employment Status
1. Full-Time
2. Part-Time
3. Intern
Your Choice? 2
```

۷. اکنون کلید Enter را بزنید.

```
========================================
==-= National Bank =-==
   Employee Record
----------------------------------------
Employee #:  42-855
First Name:  Paula
Last Name:  Meyer
Employee Title:  Cashier
Hourly Salary:  18.25
Employment Status: PartTime
========================================
```

۸. کلید Enter را زده تا از محیط Dos خارج شوید و به محیط ویژوال برگردید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

برگرداندن enumeration از متد

برای ایجاد متدی که enumeration باز می گرداند، نوع بازگشتی آن را به عنوان اسم enumeration مشخص کنید. در بدنه ی enumeration، هر کاری که لازم است انجام دهید. پیش از خروج از متد، مطمئن شوید مقداری که بازمی گردانید از نوع شمارشی (enumeration type) باشد. مثال زیر را در نظر بگیرید.

```
public class Exercise
{
  private HouseType SpecifyPropertyType()
  {
    HouseType pt;
    pt = HouseType.TownHouse;
    return pt;
  }
}
```

می توان متدی را که نوع شمارشی بازمی گرداند با اسم آن فراخواند. می توان مقدار بازگشتی آن را برای این منظور مورد استفاده قرار داد.

مثال

```
using System;
public enum HouseType
{
  Unknown = 2,
  SingleFamily = 4,
  TownHouse = 6,
  Condominium = 8,
}
public class Exercise
{
  private static void ShowHouse(HouseType propType = HouseType.Unknown)
  {
    Console.WriteLine("Type of house: {0}", propType);
  }
  private static HouseType SpecifyPropertyType()
  {
    HouseType pt;
    pt = HouseType.TownHouse;
    return pt;
  }
  public static int Main()
  {
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
        HouseType ht;
        ht = SpecifyPropertyType();
        ShowHouse(ht);
        return 0;
    }
}
```

نتیجه ی زیر حاصل می گردد.

```
Type of house: TownHouse
Press any key to continue...
```

عملگرهای منطقی

مقدمه

برنامه در واقع مجموعه ای از دستور ها است که از compiler درخواست می کند شرایط را بسنجد و بر اساس آن شرایط عمل کند. برای بررسی چنین شرایطی، کامپیوتر بخش عظیمی از زمان خود را صرف مقایسه ی مقادیر مختلف می کند. فرایند مقایسه در حقیقت همان عملیات Boolean است که بسته به مقادیری که مقایسه بر پایه ی آن انجام شده نتیجه ی صحیح (true) یا غلط (false) می دهد.

مقایسه بین دو مقدار هم نوع صورت می گیرد ؛ برای مثال، می توان دو عدد، دو کاراکتر، اسامی دو شهر را با هم مقایسه کرد. از سوی دیگر، مقایسه ی بین دو مقدار متفاوت (از نوع مختلف) کاملاً بی معنا می باشد. برای مثال، نمی توان یک شماره تلفن را با سن کسی مقایسه کرد یا یک نوع موسیقی را با فاصله ی بین دو نقطه قیاس کرد. مشابه عملیات حسابی دودویی (binary)، عملیات مقایسه بر اساس دو مقدار صورت می گیرد. بر خلاف عملیات حسابی که نتایج حاصله آن متنوع و متفاوت هستند، نتیجه ی فرایند مقایسه از دو حالت خارج نیست به طوری که نتیجه یا منطقی صحیح (logical true) یا منطقی ناصحیح (logical false) است. در صورتی که مقایسه صحیح باشد، مقدار توکار ۱ یا مثبت را به دست می دهد، که منظور همان مقداری است که از ۰ بزرگتر باشد. چنانچه مقایسه صحیح یا true نباشد، غلط یا false تلقی می گردد و مقدار آن ۰ خواهد بود.

زبان C# مجهز به عملگرهای گوناگونی است که به کمک آن ها می توان هر قسم مقاسیه ای بین مقادیر هم نوع انجام داد. مقادیر نام برده می توانند عددی، از نوع رشته و یا اشیاء باشند (عملیاتی که بر روی اشیا صورت می گیرد در فرایندی به نام operator overloading یا اضافه بارگذاری عملگر، اختصاصی تنظیم می شوند.

هنگام نوشتن دستورات شرطی (دستوراتی که در شرطی ها به کار می روند) یک سری ملزومات اولیه هست که باید رعایت کرد.

سادگی و وضوح : یک دستور باید کاملاً واضح و تا حد ممکن ساده ولی در عین حال کامل باشد. زمانی که دستوری طولانی می گردد، ممکن است به بخش های کوتاه مختلفی تقسیم شود که وضوح دستور را مختل کرده و منجر به بروز مسائل و مشکلات متعددی می شود.

واقعی : دستور مورد نظر باید به عنوان یک حقیقت عرضه شود و نه یک نظر یا عقیده، به این معنا که لزومی ندارد شما آن دستور را دوست داشته باشید بلکه باید مثل اکثریت آن را به عنوان صحیح (true) یا غلط (false) بپذیرید. در حقیقت، نیازی نیست که دستور به خودی خود درست باشد بلکه باید آن را به عنوان صحیح پذیرفت. بر این اساس، دستوری مثل اینکه " یک ساعت ۴۵ دقیقه هست " لزومی ندارد با نظر شما همخوانی داشته باشد بلکه باید یا به عنوان صحیح یا غلط پذیرفته شود. دستوری مثل " این متقاضی شغل گزینه ی مناسبی است " یک نظر می باشد، بنابراین در دستورات شرطی جایی ندارد.

درستی ضمنی و وابسته به موقعیت : زمانی که دستوری نوشته یا ایجاد می شود باید به عنوان صحیح یا غلط پذیرفته شود، اگرچه ممکن است بعده ها تغییر کند. برای مثال، تصور کنید در سال مشخصی دستوری به این صورت نوشته شود " امسال ماه فبریه ۲۸ روز خواهد داشت ". اگرچه استفاده از آن مجاز می باشد، باید تا حد ممکن از به کاربردن آن خودداری کرد، مگر در شرایط اضطراری.

معکوس : باید این امکان وجود داشته باشد که یک دستور عکس خود را پیدا کند. به عبارت روشن تر، زمانی که دستوری نوشته و به عنوان صحیح یا غلط پذیرفته می شود، دستور ضد یا مغایری باید وجود داشته باشد که آن را عکس کند (از غلط به صحیح و بالعکس). برای مثال، چنانچه دستوری دارید مانند " این متقاضی کار ۱۸ سال سن دارد " باید این امکان وجود داشته باشد که بگویید " این متقاضی کار ۱۸ ساله نیست " یا " این متقاضی کار جوان تر از ۱۸ سال به نظر می رسد ".

سعی کنید تا حد ممکن دستورات خود را دقیق و واضح بیان و تنظیم کنید. با این کار برنامه های شما خوانا می شوند و عیب زدایی آن آسان می گردد.

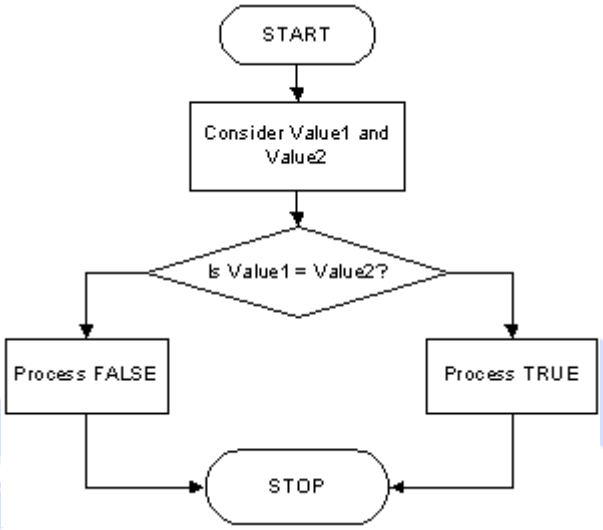
عملگر تساوی ==

برای مقایسه ی دو متغیر، C# عملگر == را به کار می برد. فرمول آن به شرح زیر می باشد.

value1 == Value2

عملیات تساوی به منظور کشف برابری بین دو متغیر (یا یک متغیر و یک ثابت) بکار می رود؛ به عبارت دیگر، ایا دو متغیر مقدار یکسانی دارند یا نه. compiler از طریق دستور نحوی، مقدار value1 و value2 را با هم مقایسه می کند. چنانچه مقدار این دو value با هم یکسان یا برابر بود، عملیات مقایسه نتیجه ی صحیح یا true را به دست می دهد، اما در صورتی که دو متغیر فوق با هم متفاوت باشند (مقدار آن ها باهم برابر نباشد)، مقایسه نتیجه ی غلط یا false را ارائه می دهد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330



اغلب مقایسه هایی که در زبان C# صورت می گیرد، برای دستورات شرطی به کار گرفته می شود. نتیجه ی پروسه ی مقایسه را می توان به یک متغیر نیز اختصاص داد. برای ذخیره ی نتیجه ی مقایسه، لازم است عملیات مقایسه را داخل پرانتز قرار دهید.

مثال

```
using System;
public class Exercise
{
  static int Main()
  {
    var value1 = 15;
    var value2 = 24;
    Console.Write("Value 1 = ");
    Console.WriteLine(value1);
    Console.Write("Value 2 = ");
    Console.WriteLine(value2);
    Console.Write("Comparison of value1 == 15 produces ");
    Console.WriteLine(value1 == 15);
    return 0;
  }
}
```

نتیجه ی زیر حاصل می گردد.

```
Value 1 = 15
Value 2 = 24
Comparison of value1 == 15 produces True
```

لازم است بین عملگر جایگزین " = " و عملگر منطقی تساوی " == " تفاوت قائل شد. عملگر جایگزین برای دادن مقداری جدید به متغیر به کار می رود مانند عدد 244 =. به یاد داشته باشید که عملوندی (operand) که در سمت چپ " = " قرار می گیرد باید همیشه یک متغیر باشد و نه یک ثابت. عملگر " == " هیچگاه برای تخصیص دادن مقدار به کار نمی رود، و در صورت استفاده از آن برای این منظور با error مواجه می شوید. عملگر " == " تنها برای مقایسه ی دو مقدار به کار می رود. عملوندی که در سمت چپ عملگر مزبور قرار می گیرد می تواند متغیر، ثابت و یا یکی متغیر و دیگری ثابت باشد. چنانچه از یک عملگر به طور اشتباه به جای دیگری استفاده کنید هنگام ترجمه ی (compile) برنامه با error مواجه می شوید.

عملگر منطقی Not

پس از تعریف و مقداردهی متغیر (این کار از طریق مقداردهی اولیه یا تغییر مقدار انجام می شود)، در واقع متغیر مذکور در برنامه زنده (یا ایجاد) می شود. حال متغیر می تواند در عملیات لازم شرکت داشته باشد. compiler حساب تمامی متغیرهای موجود در برنامه (و متغیرهایی که در حال پردازش هستند) را دارد. (چنانچه متغیری مورد استفاده قرار نگیرد یا برای پردازش در دسترس نباشد که در برنامه نویسی مجازی غیرفعال تلقی می گردد) برای غیرفعال کردن متغیر (به طور موقت)، باید مقدار آن را nullify (تخصیص مقدار null به متغیر) کرد. C# متغیری را که مقدار آن تهی (Null) باشد stern در نظر می گیرد. به منظور غیر فعال کردن (غیر قابل استفاده و از دسترس خارج ساختن آن) متغیر، حین توسعه و شکل گیری برنامه، عملگر منطقی Not " ! " را اعمال کنید. ترکیب نحوی آن به صورت زیر می باشد.

!Value

برای استفاده از عملگر منطقی Not دو گزینه ی اصلی پیش رو دارید. یکی از معمول ترین روش های استفاده از آن، بررسی وضعیت متغیر است.

برای تخصیص مقدار null به متغیر ، کافی است علامت تعجب " ! " را سمت چپ متغیر قرار دهید. مانند مثال زیر

```
using System;
public class Exercise
{
  static int Main()
  {
    bool hasAirCondition = true;
    bool doesIt;
    Console.Write("hasAirCondition = ");
    Console.WriteLine(hasAirCondition);
    doesIt = !hasAirCondition;
    Console.Write("doesIt      = ");
    Console.WriteLine(doesIt);
```

```
        return 0;
    }
}
```

این نتیجه حاصل می گردد.

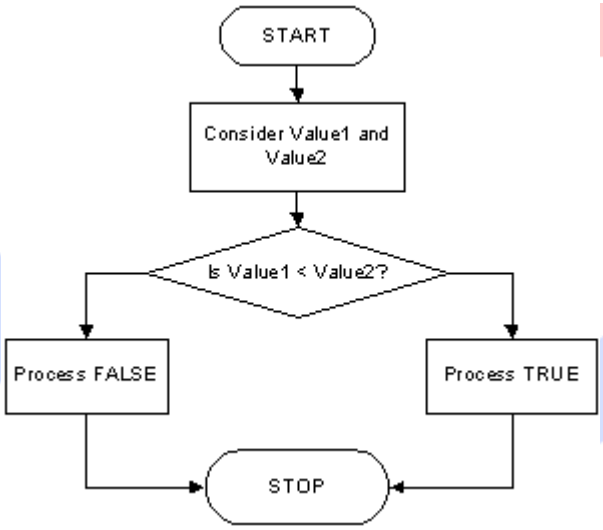
```
hasAirCondition = True
doesIt= False
```

زمانی که متغیری مقداری دارد، " زنده " یا فعال تلقی می شود. برای از دسترس خارج ساختن متغیر می توان عملگر " Not " را به آن تخصیص داد. پس از تخصیص مقدار تهی (null) به متغیر، مقدار منطقی آن تغییر می یابد. اگر مقدار منطقی متغیر true یا همان ۱ بود، حال به false یا ۰ تبدیل می شود. به این ترتیب می توان مقدار منطقی متغیر را معکوس کرد.

کوچکتر از : <

برای کوچکتر نشان دادن مقداری از مقدار دیگر، عملگر < را به کار ببرید. ترکیب نحوی عملگر نام برده به این شکل است.

Value1 < Value2

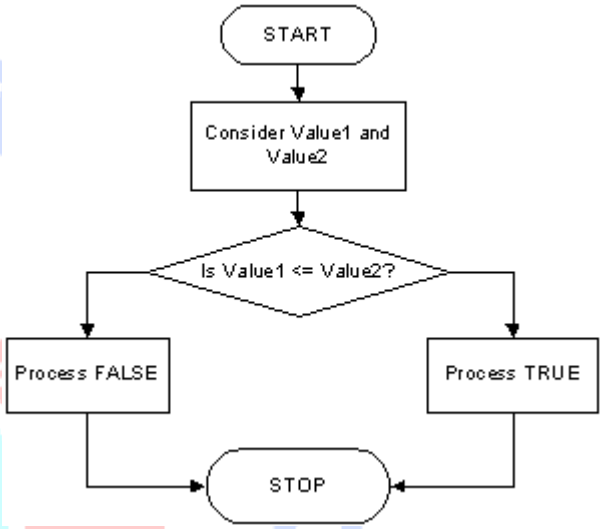


کوچکتر یا مساوی : =<

دو عملیات پیشین را می توان با هم ترکیب کرد و با استفاده از آن دو مقدار را با هم مقایسه کرد. با این کار متوجه می شوید که دو مقدار با هم برابرند یا مقدار اولی از مقدار دومی کوچک تر است. عملگر گفته شده به این شکل است : =< و دستور نحوی آن به ترتیب زیر می باشد.

Value1 <= Value2

عملگر <= درست مثل دو عمگلر پیشین، عملیات مقایسه انجام می دهد. در صورتی که مقدار value1 و value2 برابر باشد، نتیجه صحیح یا مثبت به دست می آید. چنانچه عملوند طرف چپ، که در این مورد منظور value1 است، مقداری داشته باشد که از مقدار عملوند دیگر، value2، کمتر یا کوچکتر باشد نتیجه باز هم صحیح (true/positive) خواهد بود.



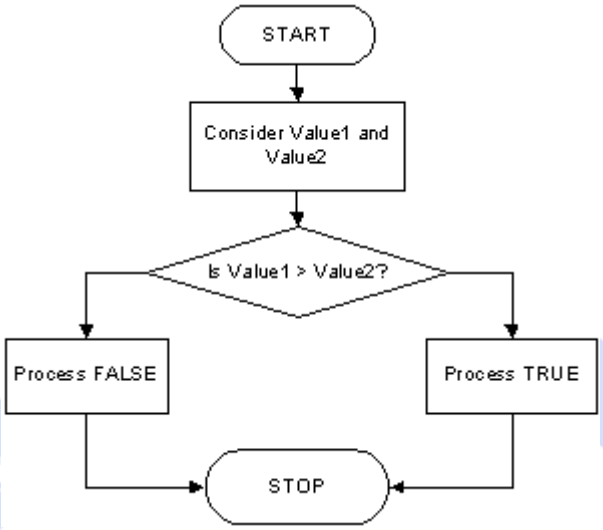
بزرگتر از: >

برای بزرگتر نشان دادن مقداری از مقدار دیگر، عملگر > مورد استفاده قرار می گیرد. فرمول آن به صورت زیر است.

Value1 > Value2

هر دو عملگر، در این مثال value1 و value2، می توانند متغیر باشند، یا عملگر سمت چپ متغیر و دیگری ثابت باشد. چنانچه مقدار واقع در سمت چپ عملگر بزرگتر از مقدار طرف راست آن بود، مقایسه نتیجه ی صحیح یا مثبت را به دست می دهد. در غیر این صورت، مقایسه نتیجه ی غلط یا تهی (null) ارائه می دهد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

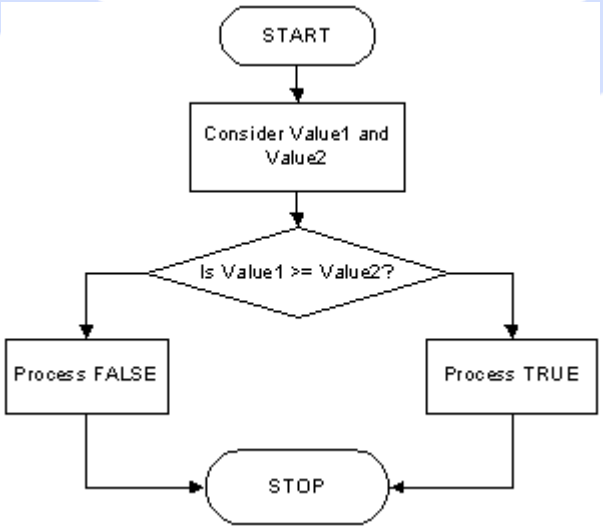


بزرگتر یا مساوی =>

دو عملگر بزرگتر از و تساوی با هم ترکیب شده و عملگر رو به رو را تولید می کند : =>. به این عملگر " بزرگتر از یا مساوی " می گویند. دستور نحوی آن به صورت زیر است.

Value1 >= Value2

عملیات مقایسه روی هر دو عملوند value1 و value2 اجرا می شود. چنانچه مقدار هر دو عملوند برابر باشد، مقایسه نتیجه ی صحیح یا مثبت می دهد. در صورتی که مقدار عملوند چپ از مقدار عملوند راست بزرگتر باشد، باز هم نتیجه صحیح (true) خواهد شد. حال اگر مقدار عملوند سمت چپ از مقدار عملوند سمت راست کوچکتر بود، نتیجه ی فرایند غلط یا تهی (null) خواهد شد.



آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

جدول زیر چکیده ی عملگرهای منطقی نام برده را فهرست می کند.

Operator	Meaning	Example	Opposite
==	Equality to	a == b	!=
!=	Not equal to	12 != 7	==
<	Less than	25 < 84	>=
<=	Less than or equal to	Cab <= Tab	>
>	Greater than	248 > 55	<=
>=	Greater than or equal to	Val1 >= Val2	<

دستورات شرطی

چنانچه شرطی درست بود

دستور شرطی، عبارتی است که نتیجه ی صحیح (true) یا غلط (false) بدهد. به منظور ایجاد عبارت (expression) لازم، عملگرهای Boolean را به کار می بریم. در مبحث پیشین، با نحوه ی اجرا عملیات و دریافت نتایج و به کاربردن آن ها آشنا شدیم. برای استفاده از عملیات Boolean، زبان برنامه نویسی C# عملگر های شرطی خود را ارائه می دهد.

عبارت های شرطی

۱. Microsoft Visual Studio را راه اندازی کنید.

۲. برای ایجاد برنامه ی جدید، به فهرست گزینه ی اصلی مراجعه کرده، روی File -> New Project... کلیک کنید.

۳. در لیست میانی روی Empty Project کلیک کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

۴. اسم پروژه را به ElectronicStore2 تغییر دهید.

۵. کلید Enter را بزنید.

۶. به منظور ایجاد کلاس جدید، در پنجره ی Class View راست کلیک کرده سپس : ElectronicStore2 -> Add -> Class...

۷. اسم کلاس را StoreItem انتخاب کرده، سپس Add را کلیک کنید.

۸. فایل StoreItem.cs را به صورت زیر اصلاح کنید.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ElectronicStore2
{
  public enum ItemsCategories
  {
    Unknown,
    CablesAndConnectors,
    CellPhonesAndAccessories,
    Headphones,
    DigitalCameras,
    PDAsAndAccessories,
    TelephonesAndAccessories,
    TVsAndVideos,
    SurgeProtectors,
    Instructional
  }
  public class StoreItem
  {
    public long itemNumber;
    public string name;
    public ItemsCategories category;
    public string manufacturer;
    public string model;
    public decimal unitPrice;
    public StoreItem(long itmNbr = 0, string itemName = "N/A",
            ItemsCategories kind = ItemsCategories.Unknown,
            string make = "Unknown", string mdl = "Unspecified",
            decimal price = 0.00M)
    {
      itemNumber = itmNbr;
      name = itemName;
      category = kind;
      manufacturer = make;
      model = mdl;
      unitPrice = price;
    }
  }
```

```
}
```

ایجاد شرط if

برنامه زیر را در نظر بگیرید:

```
using System;
  public enum HouseType
  {
    Unknown,
    SingleFamily,
    TownHouse,
    Condominium
  }
  public class Exercise
  {
    static int Main()
    {
      var type = HouseType.Unknown;
      var choice = 0;
      Console.WriteLine("Enter the type of house you want to purchase");
      Console.WriteLine("1. Single Family");
      Console.WriteLine("2. Townhouse");
      Console.WriteLine("3. Condominium");
      Console.Write("You Choice? ");
      choice = int.Parse(Console.ReadLine());
      Console.WriteLine("\nDesired House Type: {0}", type);
      return 0;
    }
  }
```

نتیجه

```
Enter the type of house you want to purchase
1. Single Family
2. TownHouse
3. Condominium
You Choice? 3
Desired House Type: Unknown
Press any key to continue...
```

به منظور بررسی عبارت (اینکه آیا صحیح true/ است یا نه) و استفاده از نتیجه ی Boolean آن، عملگر if به کار برده می شود. فرمول آن به شرح زیر است.

if(Condition) Statement;

Condition می تواند از همان نوع عملیات بولی باشد که در درس پیش با آن شنا شدیم، به عبارت دیگر می تواند فرمول زیر را داشته باشد.

Operand1 BooleanOperator Operand2

چنانچه Condition نتیجه ی صحیح تولید کند، برنامه ی مفسر، Statement را اجرا می کند. اگر دستور مربوطه (دستوری که باید اجرا شود) کوتاه بود، می توان آن را در خطی که شرط در آن بررسی می شود جای داد.

مثال

```
using System;
public enum HouseType
{
   Unknown,
   SingleFamily,
   TownHouse,
   Condominium
}
public class Exercise
{
   static int Main()
   {
      var type = HouseType.Unknown;
      var choice = 0;
      Console.WriteLine("Enter the type of house you want to purchase");
      Console.WriteLine("1. Single Family");
      Console.WriteLine("2. Townhouse");
      Console.WriteLine("3. Condominium");
      Console.Write("You Choice? ");
      choice = int.Parse(Console.ReadLine());
      if (choice == 1) type = HouseType.SingleFamily;
      Console.WriteLine("\nDesired House Type: {0}", type);
      return 0;
   }
}
```

نتیجه

```
Enter the type of house you want to purchase
1. Single Family
2. Townhouse
3. Condominium
You Choice? 1

Desired House Type: SingleFamily
Press any key to continue...
```

در صورتی که Statement طولانی بود، باید آن را در خطی مجزا از عبارت شرط if نوشت یا جای گذاری کرد. مثال

```
using System;
public enum HouseType
{
   Unknown,
   SingleFamily,
   TownHouse,
   Condominium
}
public class Exercise
{
   static int Main()
   {
      var type = HouseType.Unknown;
      var choice = 0;
      Console.WriteLine("Enter the type of house you want to purchase");
      Console.WriteLine("1. Single Family");
      Console.WriteLine("2. Townhouse");
      Console.WriteLine("3. Condominium");
      Console.Write("You Choice? ");
      choice = int.Parse(Console.ReadLine());
      if (choice == 1)
      type = HouseType.SingleFamily;
      Console.WriteLine("\nDesired House Type: {0}", type);
      return 0;
   }
}
```

همچنین می توان Statement را در خطی مجزا از عبارت شرط قرار داد حتی اگر دستور مربوط آنقدر کوتاه باشد که با شرط if در تنها یک خط جا شود.

اگر چه دستور (ساده ی) if اغلب برای بررسی تنها یک شرط به کار می رود، می توان از آن برای اجرای چندین دستور وابسته و تابع استفاده کرد. برای بررسی چندین دستور، مجموعه دستورات را داخل کاراکتر باز "{" و بسته ی "}" محصور کنید. به مثال زیر توجه کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
using System;
  public enum HouseType
  {
    Unknown,
    SingleFamily,
    TownHouse,
    Condominium
  }
  public class Exercise
  {
    static int Main()
    {
      var type = HouseType.Unknown;
      var choice = 0;
      Console.WriteLine("Enter the type of house you want to purchase");
      Console.WriteLine("1. Single Family");
      Console.WriteLine("2. Townhouse");
      Console.WriteLine("3. Condominium");
      Console.Write("You Choice? ");
      choice = int.Parse(Console.ReadLine());
      if (choice == 1)
      {
        type = HouseType.SingleFamily;
        Console.WriteLine("\nDesired House Type: {0}", type);
      }
      return 0;
    }
  }
```

چنانچه کاراکتر {} را حذف کنید، فقط دستوری که بلافاصله پس از شرط قرار گرفته اجرا می شود. همچنین می توان برای ایجاد دستور شرطی if، روی بخش مورد نظر (بخشی که مایلید کد در آن اضافه شود) راست کلیک کرده، گزینه ی Code Snippet... را انتخاب کنید. حال، روی Visual C# دوبار کلیک کنید. در لیست مورد نظر، روی if کلیک کنید.

همچنین می توان چندین شرط if ایجاد کرد. به مثال های زیر توجه کنید.

```
    using System;
  public enum HouseType
  {
    Unknown,
    SingleFamily,
    Townhouse,
    Condominium
  }
  public class Exercise
  {
    static int Main()
    {
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
        var type = HouseType.Unknown;
        var choice = 0;
        Console.WriteLine("Enter the type of house you want to purchase");
        Console.WriteLine("1. Single Family");
        Console.WriteLine("2. Townhouse");
        Console.WriteLine("3. Condominium");
        Console.Write("You Choice? ");
        choice = int.Parse(Console.ReadLine());
        if (choice == 1)
          type = HouseType.SingleFamily;
        if (choice == 2)
          type = HouseType.Townhouse;
        if (choice == 3)
          type = HouseType.Condominium;
        Console.WriteLine("\nDesired House Type: {0}", type);
        return 0;
    }
  }
```

نمونه ای از برنامه

```
Enter the type of house you want to purchase
1. Single Family
2. Townhouse
3. Condominium
You Choice? 3
Desired House Type: Condominium
Press any key to continue...
```

به کاربردن شرط ساده ی if

۱. برای ایجاد فایل جدید، فهرست گزینه ی اصلی را باز کرده، روی Project -> Add New Item... کلیک کنید.

۲. در لیست میانی، گزینه ی Code File را انتخاب کنید.

۳. اسم فایل را به ElectronicStore را تغییر داده، کلید Enter را بزنید.

۴. در فایل خالی که در اختیار شما قرار می گیرد، دستورات زیر را وارد کنید.

```
using System;
  using ElectronicStore2;
  public class Store
  {
    StoreItem CreateStoreItem()
    {
      int? category = null;
      StoreItem sItem = new StoreItem();
      Console.Title = "Electronic Super Store";
```

```
    Console.WriteLine("=-= Nearson Electonics =-=");
    Console.WriteLine("******* Store Items ******");
    Console.WriteLine(
    "To create a store item, enter its information");
    Console.Write("Item Number: ");
    sItem.itemNumber = long.Parse(Console.ReadLine());
    Console.WriteLine("Category");
    Console.WriteLine("1.  Unknown/Miscellaneous");
    Console.WriteLine("2.  Cables and Connectors");
    Console.WriteLine("3.  Cell Phones and Accessories");
    Console.WriteLine("4.  Headphones");
    Console.WriteLine("5.  Digital Cameras");
    Console.WriteLine("6.  PDAs and Accessories");
    Console.WriteLine("7.  Telephones and Accessories");
    Console.WriteLine("8.  TVs and Videos - Plasma / LCD");
    Console.WriteLine("9.  Surge Protector");
    Console.WriteLine("10. Instructional and Tutorials (VHS & DVD)TVs and Videos");
    Console.Write("Your Choice? ");
    category = int.Parse(Console.ReadLine());
    if (category == 1)
      sItem.category = ItemsCategories.Unknown;
    if (category == 2)
      sItem.category = ItemsCategories.CablesAndConnectors;
    if (category == 3)
      sItem.category = ItemsCategories.CellPhonesAndAccessories;
    if (category == 4)
      sItem.category = ItemsCategories.Headphones;
    if (category == 5)
      sItem.category = ItemsCategories.DigitalCameras;
    if (category == 6)
      sItem.category = ItemsCategories.PDAsAndAccessories;
    if (category == 7)
      sItem.category = ItemsCategories.TelephonesAndAccessories;
    if (category == 8)
      sItem.category = ItemsCategories.TVsAndVideos;
    if (category == 9)
      sItem.category = ItemsCategories.SurgeProtectors;
    if (category == 10)
      sItem.category = ItemsCategories.Instructional;
    Console.Write("Make:      ");
    sItem.manufacturer = Console.ReadLine();
    Console.Write("Model:     ");
    sItem.model = Console.ReadLine();
    Console.Write("Unit Price: ");
    sItem.unitPrice = decimal.Parse(Console.ReadLine());
    return sItem;
}
void DescribeStoreItem(StoreItem item)
{
    Console.Title = "Electronic Super Store";
```

```
        Console.WriteLine("=-= Nearson Electonics =-=");
        Console.WriteLine("******* Store Items ******");
        Console.WriteLine("Store Item Description");
        Console.WriteLine("Item Number: {0}", item.itemNumber);
        Console.WriteLine("Category:    {0}", item.category);
        Console.WriteLine("Make:        {0}", item.manufacturer);
        Console.WriteLine("Model:       {0}", item.model);
        Console.WriteLine("Unit Price:  {0:C}", item.unitPrice);
    }
    public static int Main()
    {
        Store st = new Store();
        StoreItem saleItem = st.CreateStoreItem();
        Console.Clear();
        st.DescribeStoreItem(saleItem);
        System.Console.ReadKey();
        return 0;
    }
}
```

۵. به منظور اجرای برنامه، در فهرست گزینه ی اصلی، روی Debug -> Start Debugging کلیک کنید.

۶. برای Item Number، 972485 را وارد کنید سپس Enter را بزنید.

۷. برای گزینه ی your choice عدد ۵ را وارد کنید.

۸. برای آیتم make (ساخت)، Canon را وارد کنید، سپس کلید Enter را بزنید.

۹. برای آیتم model، PowerShot SX20IS را تایپ کرده و Enter را بزنید.

۱۰. برای گزینه ی unit price (قیمت محصول انتخابی)، 369.00 تایپ کرده، اکنون Enter را بزنید.

```
=-= Nearson Electonics =-=
******* Store Items ******
To create a store item, enter its information
Item Number: 972485
Category
1. Unknown/Miscellaneous
2. Cables and Connectors
3. Cell Phones and Accessories
4. Headphones
5. Digital Cameras
6. PDAs and Accessories
7. Telephones and Accessories
8. TVs and Videos - Plasma / LCD
9. Surge Protector
10. Instructional and Tutorials (VHS & DVD)TVs and Videos
```

```
Your Choice? 5
Make:    Canon
Model:   PowerShot SX20IS
Unit Price: 369.00
```

۱۱. کلید Enter را بزنید.

```
=-= Nearson Electonics =-=
******* Store Items ******
Store Item Description
Item Number: 972485
Category:  DigitalCameras
Make:    Canon
Model:   PowerShot SX20IS
Unit Price: $369.00
Press any key to continue...
```

if...else

در زیر نمونه ای از آنچه در بخش پیشین آموختیم تشریح شده است:

```
using System;
public enum HouseType
{
  Unknown,
  SingleFamily,
  Townhouse,
  Condominium
}
public class Exercise
{
  static int Main()
  {
    var type = HouseType.Unknown;
    var choice = 0;
    Console.WriteLine("Enter the type of house you want to purchase");
    Console.WriteLine("1. Single Family");
    Console.WriteLine("2. Townhouse");
    Console.WriteLine("3. Condominium");
    Console.Write("You Choice? ");
    choice = int.Parse(Console.ReadLine());
    if (choice == 1)
      type = HouseType.SingleFamily;
    if (choice == 2)
```

```
        type = HouseType.Townhouse;
      if (choice == 3)
        type = HouseType.Condominium;
      Console.WriteLine("\nDesired House Type: {0}", type);
      if (type == HouseType.SingleFamily)
        Console.WriteLine("\nDesired House Matched");
      return 0;
    }
  }
```

چنانچه برای اجرای عملیات از شرط if استفاده بکنید ونتیجه ی آن صحیح (true) درآید، با توجه به گفته های پیشین می توان دستور مربوطه را اجرا کرد. همان طور که در بخش پیشین به آن پرداختیم، هر نتیجه ی دیگری نادیده گرفته می شود. به عنوان یک جایگزین برای شرط if در چنین مواقعی از else استفاده می شود (به این معنا که اگر if اجرا نشد else اجرا می شود). فرمول آن به ترتیب زیر است:

```
if(Condition)
    Statement1;
  else
    Statement2;
```

به خاطر داشته باشید این بار هم Condition می تواند یک عملیات Boolean باشد، مشابه آنچه در مبحث پیشین به آن پرداختیم. در صورتی که Condition درست بود، کامپایلر Statement1 را اجرا می کند. چنانچه Condition غلط (false) بود، کامپایلر Statement2 را اجرا می کند.

مثال

```
using System;
public enum HouseType
{
  Unknown,
  SingleFamily,
  Townhouse,
  Condominium
}
public class Program
{
  static int Main()
  {
    var type = HouseType.Unknown;
    var choice = 0;
    Console.WriteLine("Enter the type of house you want to purchase");
    Console.WriteLine("1. Single Family");
    Console.WriteLine("2. Townhouse");
    Console.WriteLine("3. Condominium");
```

```
      Console.Write("You Choice? ");
      choice = int.Parse(Console.ReadLine());
      if (choice == 1)
        type = HouseType.SingleFamily;
      if (choice == 2)
        type = HouseType.Townhouse;
      if (choice == 3)
        type = HouseType.Condominium;
      Console.WriteLine("\nDesired House Type: {0}", type);
      if (type == HouseType.SingleFamily)
        Console.WriteLine("Desired House Matched");
      else
        Console.WriteLine("No House Desired");
      return 0;
    }
  }
```

نمونه ای از نتیجه اجرای برنامه

```
Enter the type of house you want to purchase
 1. Single Family
 2. Townhouse
 3. Condominium
 You Choice? 1
 Desired House Type: SingleFamily
 Desired House Matched
 Press any key to continue...
```

نمونه ای دیگر

```
Enter the type of house you want to purchase
 1. Single Family
 2. Townhouse
 3. Condominium
 You Choice? 2
 Desired House Type: Townhouse
 No House Desired
 Press any key to continue...
```

استفاده از شرط if...else

۱. به منظور استفاده از شرط if...else فایل ElectronicStore.cs را به ترتیب زیر اصلاح کنید.

```
using System;
using ElectronicStore2;
public class Store
{
  StoreItem CreateStoreItem()
  {
    int? category = null;
    decimal itemPrice = 0m;
    StoreItem sItem = new StoreItem();
    Console.Title = "Electronic Super Store";
    Console.WriteLine("=-= Nearson Electonics =-=");
    Console.WriteLine("******* Store Items ******");
    Console.WriteLine(
    "To create a store item, enter its information");
    Console.Write("Item Number: ");
    sItem.itemNumber = long.Parse(Console.ReadLine());
    Console.WriteLine("Category");
    Console.WriteLine("1. Unknown/Miscellaneous");
    Console.WriteLine("2. Cables and Connectors");
    Console.WriteLine("3. Cell Phones and Accessories");
    Console.WriteLine("4. Headphones");
    Console.WriteLine("5. Digital Cameras");
    Console.WriteLine("6. PDAs and Accessories");
    Console.WriteLine("7. Telephones and Accessories");
    Console.WriteLine("8. TVs and Videos - Plasma / LCD");
    Console.WriteLine("9. Surge Protector");
    Console.WriteLine("10. Instructional and Tutorials (VHS & DVD)TVs and Videos");
    Console.Write("Your Choice? ");
    category = int.Parse(Console.ReadLine());
    if (category == 1)
      sItem.category = ItemsCategories.Unknown;
    if (category == 2)
      sItem.category = ItemsCategories.CablesAndConnectors;
    if (category == 3)
      sItem.category = ItemsCategories.CellPhonesAndAccessories;
    if (category == 4)
      sItem.category = ItemsCategories.Headphones;
    if (category == 5)
      sItem.category = ItemsCategories.DigitalCameras;
    if (category == 6)
      sItem.category = ItemsCategories.PDAsAndAccessories;
    if (category == 7)
      sItem.category = ItemsCategories.TelephonesAndAccessories;
    if (category == 8)
      sItem.category = ItemsCategories.TVsAndVideos;
    if (category == 9)
      sItem.category = ItemsCategories.SurgeProtectors;
    if (category == 10)
      sItem.category = ItemsCategories.Instructional;
    Console.Write("Make:      ");
```

```
    sItem.manufacturer = Console.ReadLine();
    Console.Write("Model:     ");
    sItem.model = Console.ReadLine();
    Console.Write("Unit Price: ");
    itemPrice = decimal.Parse(Console.ReadLine());
    if (itemPrice <= 0)
      sItem.unitPrice = 0.00m;
    else
      sItem.unitPrice = itemPrice;
    return sItem;
}
string GetItemCategory(ItemsCategories cat)
{
    string strCategory = "Unknown";
    if (cat == ItemsCategories.CablesAndConnectors)
      strCategory = "Cables & Connectors";
    if (cat == ItemsCategories.CellPhonesAndAccessories)
      strCategory = "Cell Phones & Accessories";
    if (cat == ItemsCategories.Headphones)
      strCategory = "Headphones";
    if (cat == ItemsCategories.DigitalCameras)
      strCategory = "Digital Cameras";
    if (cat == ItemsCategories.PDAsAndAccessories)
      strCategory = "PDAs & Accessories";
    if (cat == ItemsCategories.TelephonesAndAccessories)
      strCategory = "Telephones & Accessories";
    if (cat == ItemsCategories.TVsAndVideos)
      strCategory = "TVs & Videos";
    if (cat == ItemsCategories.SurgeProtectors)
      strCategory = "Surge Protectors";
    if (cat == ItemsCategories.Instructional)
      strCategory = "Instructional";
    return strCategory;
}
void DescribeStoreItem(StoreItem item)
{
    string strCategory = GetItemCategory(item.category);
    Console.Title = "Electronic Super Store";
    Console.WriteLine("=-= Nearson Electonics =-=");
    Console.WriteLine("******* Store Items ******");
    Console.WriteLine("Store Item Description");
    Console.WriteLine("Item Number: {0}", item.itemNumber);
    Console.WriteLine("Category:    {0}", strCategory);
    Console.WriteLine("Make:        {0}", item.manufacturer);
    Console.WriteLine("Model:       {0}", item.model);
    Console.WriteLine("Unit Price:  {0:C}", item.unitPrice);
}
public static int Main()
{
    Store st = new Store();
```

```
        StoreItem saleItem = st.CreateStoreItem();
        Console.Clear();
        st.DescribeStoreItem(saleItem);
        System.Console.ReadKey();
        return 0;
    }
}
```

۲. برای اجرای برنامه, فهرست گزینه ی اصلی را باز کرده, سپس : Debug -> Start Debugging

۳. حال مقادیر را به ترتیب زیر وارد کنید و پس از هر کدام کلید Enter را فشار دهید.

```
=-= Nearson Electonics =-=
******* Store Items ******
To create a store item, enter its information
Item Number: 992052
Category
1. Unknown/Miscellaneous
2. Cables and Connectors
3. Cell Phones and Accessories
4. Headphones
5. Digital Cameras
6. PDAs and Accessories
7. Telephones and Accessories
8. TVs and Videos - Plasma / LCD
9. Surge Protector
10. Instructional and Tutorials (VHS & DVD)TVs and Videos
Your Choice? 4
Make:     Sennheiser
Model:    HD-555
Unit Price: 104.00
```

۴. Enter را بزنید.

```
=-= Nearson Electonics =-=
******* Store Items ******
Store Item Description
Item Number:  992052
Category:     Headphones
Make:         Sennheiser
Model:        HD-555
Unit Price:   $104.00
```

۵. اکنون پنجره ی DOS را بسته و به محیط برنامه نویسی برگردید.

عملگر های if...else

عملگر سه تایی (:?)

چنانچه شرطی دارید که ابتدا به عنوان if بررسی شود در غیر این صورت به عنوان else (اگر if غلط بود و اجرا نشد، else یا دستور دوم اجرا شد)، می توان از عملگر سه تایی استفاده کرد که ترکیبی از " ? " و " : " می باشد.

فرمول

```
Condition ? Statement1 : Statement2;
```

کامپایلر ابتدا Condition را چک می کند. اگر Condition درست بود Statement1 اجرا می شود، در غیر این صورت Statement2 اجرا می شود. برنامه ی زیر با استفاده از عملگر شرطی دو عدد را باهم مقایسه کرده و عدد بزرگ تر را مشخص می کند.

```
using System;
public class Exercise
{
  static int Main()
  {
    var Number1 = 0;
    var Number2 = 0;
    var Maximum = 0;
    var Num1 = "";
    var Num2 = "";
    Console.Write("Enter first numbers: ");
    Num1 = Console.ReadLine();
    Console.Write("Enter second numbers: ");
    Num2 = Console.ReadLine();
    Number1 = int.Parse(Num1);
    Number2 = int.Parse(Num2);
    Maximum = (Number1 < Number2) ? Number2 : Number1;
    Console.Write("\nThe maximum of ");
    Console.Write(Number1);
    Console.Write(" and ");
    Console.Write(Number2);
    Console.Write(" is ");
    Console.WriteLine(Maximum);
    Console.WriteLine();
    return 0;
  }
}
```

مثال

```
Enter first numbers: 244
Enter second numbers: 68
The maximum of 244 and 68 is 244
```

عملگر " ?? "

به خاطر دارید که هنگام تعریف متغیر (از) نوع اولیه، می توان یک علامت سوال به نوع داده اضافه کرد که از این طریق مشخص شود متغیر مذکور می تواند مقدار nullable داشته باشد.

مثال

```
using System;
public class Exercise
{
  public static int Main()
  {
    double? distance = null;
    Console.WriteLine();
    return 0;
  }
}
```

ممکن است لازم شود مقدار چنین متغیری را به متغیر دیگری تخصیص دهید. چنانچه مقدار متغیر null باشد (به این معنا که مقداری نداشته باشد یا مقدار آن تهی باشد)، دیگر تخصیص آن به متغیر دیگر بی معنا خواهد بود. بنابراین، ابتدا باید بررسی شود که مقدار متغیر مزبور، مقداری تهی است یا مقدار واقعی. به عبارت دیگر از compiler درخواست بررسی متغیر را می کنید تا مشخص شود مقدار آن مقداری تهی است یا مقداری واقعی.

در صورتی که متغیر مقداری حقیقی دارد، نه تهی، می توان مقدار آن را به متغیر جدید تخصیص داد

چنانچه متغیر تهی (null) بود، مقداری جایگزین به متغیر نام برده اختصاص می دهید

برای پشتیبانی از این امکان، زبان C# عملگر "??" را در اختیار شما قرار می دهد. فرمول آن به ترتیب زیر است.

```
TargetVariable = OriginalVariable ?? AlternateValue;
```

در وهله ی اول لازم است OriginalVariable را تعریف کرده و این متغیر باید قادر به نگه داشتن مقدار تهی باشد که از طریق افزودن عملگر ? به آن امکان پذیر می شود. توجه داشته باشید که هم می توان TargetVariable را پیش از مقداردهی اولیه تعریف کرد و هم حین مقداردهی آن. لازم به ذکر است که هر دو متغیر باید سازگار با هم باشند. مثال ذیل عملگر ?? را به کار می برد.

```
using System;
public class Exercise
{
  public static int Main()
  {
    double? distance = null;
    double? fromTo = null;
    Console.WriteLine("Distance 1: {0}", distance);
    Console.WriteLine("Distance 2: {0}", fromTo);
    fromTo = distance ?? 135.85;
    Console.WriteLine("Distance 1: {0}", distance);
    Console.WriteLine("Distance 2: {0}", fromTo);
    Console.WriteLine();
    return 0;
  }
}
```

دستور fromTo = distance ?? 135.85; دو مفهوم زیر را می رساند.

اگر متغیر distance در حال حاضر مقدار غیر تهی (non-null) دارد، مقدار مورد نظر را به متغیر fromTo تخصیص دهید.

اما اگر متغیر distance در حال حاضر مقداری تهی دارد، مقدار 135.85 را به آن تخصیص دهید

کد بالا نتیجه ی زیر را تولید می کند.

```
Distance 1:
Distance 2:
Distance 1:
Distance 2: 135.85
Press any key to continue...
```

نسخه ی دیگر برنامه

```
using System;
public class Exercise
{
  public static int Main()
  {
    double? distance = null;
    double? fromTo = null;
    Console.WriteLine("Distance 1: {0}", distance);
    Console.WriteLine("Distance 2: {0}", fromTo);
```

```
        Console.WriteLine("----------------------");
        fromTo = distance ?? 135.85;
        Console.WriteLine("Distance 1: {0}", distance);
        Console.WriteLine("Distance 2: {0}", fromTo);
        Console.WriteLine("----------------------");
        distance = 8284.26;
        fromTo = distance ?? 135.85;
        Console.WriteLine("Distance 1: {0}", distance);
        Console.WriteLine("Distance 2: {0}", fromTo);
        Console.WriteLine();
        return 0;
    }
}
```

این بار برنامه نتیجه ی زیر را ارائه می دهد.

```
Distance 1:
Distance 2:
----------------------
Distance 1:
Distance 2: 135.85
----------------------
Distance 1: 8284.26
Distance 2: 8284.26
Press any key to continue...
```

if...else if و if...else

در صورت استفاده از دستور شرطی if...else، می توان تنها دو دستور را پردازش کرد. در برخی موارد، لازم است چندین (بیش ازدو) دستور را پردازش یا اجرا کنیم. برای این منظور می توان از شرط if...else if استفاده کرد. فرمول آن به ترتیب زیر می باشد.

```
if(Condition1) Statement1;
    else if(Condition2) Statement2
```

کامپایلر ابتدا Condition1 را بررسی می کند. در صورت صحیح (true) بودن Condition1، Statement1 اجرا می شود. چنانچه Condition1 غلط (false) بود، کامپایلر Condition2 را چک می کند. در صورت درست بودن Condition2، کامپایلر Statement2 را اجرا می کند. هر نتیجه ی دیگری نادیده گرفته می شود. توجه خود را به مثال زیر جلب کنید.

```
using System;
public enum HouseType
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
{
  Unknown,
  SingleFamily,
  Townhouse,
  Condominium
}
public class Exercise
{
  static int Main()
  {
    var type = HouseType.Unknown;
    var choice = 0;
    var garage = "";
    Console.WriteLine("Enter the type of house you want to purchase");
    Console.WriteLine("1. Single Family");
    Console.WriteLine("2. Townhouse");
    Console.WriteLine("3. Condominium");
    Console.Write("You Choice? ");
    choice = int.Parse(Console.ReadLine());
    if (choice == 1) type = HouseType.SingleFamily;
    else if (choice == 2) type = HouseType.Townhouse;
    Console.Write("Does the house have an indoor garage (1=Yes/0=No)? ");
    var answer = int.Parse(Console.ReadLine());
    if (answer == 1)
      garage = "Yes";
    else
      garage = "No";
    Console.WriteLine("\nDesired House Type: {0}", type);
    Console.WriteLine("Has indoor garage? {0}", garage);
    return 0;
  }
}
```

نمونه ای از اجرای برنامه

```
Enter the type of house you want to purchase
1. Single Family
2. Townhouse
3. Condominium
You Choice? 1
Does the house have an indoor garage (1=Yes/0=No)? 1
 Desired House Type: SingleFamily
Has indoor garage?  Yes
Press any key to continue...
```

نمونه ای دیگری از اجرای برنامه

```
Enter the type of house you want to purchase
 1. Single Family
 2. Townhouse
 3. Condominium
 You Choice? 2
 Does the house have an indoor garage (1=Yes/0=No)? 6
 Desired House Type: Townhouse
 Has indoor garage?  No
 Press any key to continue...
```

همان طور که مشاهده می کنید تنها دو شرط ارزیابی شده اند. هر شرط دیگری غیر از این دو نادیده گرفته می شود. زبان C# خود به عنوان آخرین راه حل یک else جایگزین در نظر می گیرد. فرمول آن به شرح زیر است.

```
if( Condition1)
   Statement1;
else if( Condition2)
   Statement2;
else
   Statement-n;
```

```
if( Condition1)
   Statement1;
else if( Condition2)
   Statement2;
else if( Condition3)
   Statement3;
else
   Statement-n;
```

Compiler اولین شرط را بررسی می کند. در صورت درست بودن Condition1، Statement1 اجرا می شود. چنانچه Condition2 صحیح باشد، Statement2 اجرا می شود. زمانی که کامپایلر Condition-n را به true تفسیر کند، دستور همخوان (مربوطه ی) آن اجرا می شود. حال اگر Condition-n غلط باشد، کامپایلر شرط بعدی را بررسی می کند. به عبارت دیگر، می توان با استفاده از دستور else if هر تعداد شرط

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

که لازم است شامل کرد. اگر پس از بررسی جمیع شرایط احتمالی، باز هم فکر می کنید ممکن است شرطی غیر منتظره وجود داشته باشد، می توانید از else کمک بگیرید.

مثال

```
using System;
public enum HouseType
{
   Unknown,
   SingleFamily,
   Townhouse,
   Condominium
}
public class Exercise
{
   public static int Main()
   {
      var type = HouseType.Unknown;
      var choice = 0;
      var garage = "";
      Console.WriteLine("Enter the type of house you want to purchase");
      Console.WriteLine("1. Single Family");
      Console.WriteLine("2. Townhouse");
      Console.WriteLine("3. Condominium");
      Console.Write("You Choice? ");
      choice = int.Parse(Console.ReadLine());
      if (choice == 1)
         type = HouseType.SingleFamily;
      else if (choice == 2)
         type = HouseType.Townhouse;
      else if (choice == 3)
         type = HouseType.Condominium;
      else
         type = HouseType.Unknown;
      Console.Write("Does the house have an indoor garage (1=Yes/0=No)? ");
      var answer = int.Parse(Console.ReadLine());
      if (answer == 1)
         garage = "Yes";
      else
         garage = "No";
      Console.WriteLine("\nDesired House Type: {0}", type);
      Console.WriteLine("Has indoor garage? {0}", garage);
      return 0;
   }
}
```

زیر مثالی از اجرای برنامه را مشاهده می کنید.

```
Enter the type of house you want to purchase
 1. Single Family
 2. Townhouse
 3. Condominium
 You Choice? 3
 Does the house have an indoor garage (1=Yes/0=No)? 0
 Desired House Type: Condominium
 Has indoor garage?  No
 Press any key to continue...
```

دستورهای شرطی Switch

ساختار شرطی Case switch

هنگام تعریف عبارت (که نتیجه ی آن به اجرای برنامه ی خاصی منجر می شود)، دستور switch نتیجه را ارزیابی و بررسی می کند و براساس نتیجه ی احتمالی آن عبارت، دستور را اجرا می کند. به این نتیجه ی احتمالی case می گویند.

نتایج حاصله در بدنه ی اصلی دستور switch فهرست شده و هر case (نتیجه ی ممکن) در صورت لزوم باید جداگانه اجرا شود. بدنه ی اصلی دستور switch داخل کاراکترهای " } " و " { " محصور می شود. ترکیب نحوی دستور switch به صورت زیر می باشد.

```
switch(Expression)
{
  case Choice1:
     Statement1;
  break;
  case Choice2:
     Statement2;
  break;
  case Choice-n:
     Statement-n;
```

```
  break;
}
```

در زبان C++، می توان کلید واژه ی break را از case حذف کرد. به عبارت دقیق تر، پس از اینکه کد مورد نظر در case اجرا می شود به دلیل عدم وجود break، case بعدی اجرا می شود که منجر به سردرگمی و بروز مشکلات در عملیات می شود. اما در زبان برنامه نویسی C#، به منظور جلوگیری از بروز این رخداد، لازم است انتهای هر case وقفه در کد ایجاد شود. پروسه ی ایجاد وقفه با استفاده از کلید واژه ی break امکان پذیر می باشد.

عبارتی که در دستور case مورد بررسی قرار می گیرد یک عدد صحیح یا integer می باشد. به این خاطر که عضو شمارنده (enum) و نوع داده های کاراکتر مورد نظر (char) فرم های دیگر integer هستند، از آن ها نیز می توان استفاده کرد. در مثال زیر دستور switch مورد استفاده قرار گرفته است.

```
using System;
public enum HouseType
{
  Unknown,
  SingleFamily,
  Townhouse,
  Condominium
}
public class Exercise
{
  public static int Main()
  {
    var type = HouseType.Unknown;
    var choice = 0;
    var garage = "";
    Console.WriteLine("Enter the type of house you want to purchase");
    Console.WriteLine("1. Single Family");
    Console.WriteLine("2. Townhouse");
    Console.WriteLine("3. Condominium");
    Console.Write("You Choice? ");
    choice = int.Parse(Console.ReadLine());
    switch (choice)
    {
      case 1:
        type = HouseType.SingleFamily;
        break;
      case 2:
        type = HouseType.Townhouse;
        break;
      case 3:
        type = HouseType.Condominium;
        break;
```

```
        }
        Console.Write("Does the house have an indoor garage (1=Yes/0=No)? ");
        var answer = int.Parse(Console.ReadLine());
        if (answer == 1)
          garage = "Yes";
        else
          garage = "No";
        Console.WriteLine("\nDesired House Type: {0}", type);
        Console.WriteLine("Has indoor garage? {0}", garage);
        return 0;
      }
    }
```

هنگام تصدیق نتایج احتمالی که دستور switch بررسی می کند، گاهی احتمالاتی جز آنچه فهرست شده وجود دارد. case مزبور با کلیدواژه ی default مدیریت می شود. case پیش فرض (default case) زمانی لحاظ می شود که هیچ یک از case های فهرست شده با پاسخ یا نتیجه ی عرضه شده همخوانی یا تطابق نداشته باشد. ترکیب نحوی دستور switch که case پیش فرض را مورد بررسی قرار می دهد به ترتیب زیر می باشد.

```
switch(Expression)
{
  case Choice1:
     Statement1;
  break;
  case Choice2:
     Statement2;
  break;
  case Choice-n:
     Statement-n;
  break;
  default:
     Other-Possibility;
  break;
}
```

در C++، بخش default به این خاطر که آخرین قسمت switch محسوب می شود نیازی به کلید واژه ی break نیست. اما در زبان C#، هر case و بخش default باید مکانیزم خروج خود را داشته باشد که با کلید واژه ی break انجام می پذیرد.

بنابراین، ورژن دیگر برنامه ی بالا به صورت زیر می باشد.

```
using System;
public enum HouseType
{
  Unknown,
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    SingleFamily,
    Townhouse,
    Condominium
}
public class Exercise
{
    public static int Main()
    {
        var type = HouseType.Unknown;
        var choice = 0;
        var garage = "";
        Console.WriteLine("Enter the type of house you want to purchase");
        Console.WriteLine("1. Single Family");
        Console.WriteLine("2. Townhouse");
        Console.WriteLine("3. Condominium");
        Console.Write("You Choice? ");
        choice = int.Parse(Console.ReadLine());
        switch (choice)
        {
            case 1:
                type = HouseType.SingleFamily;
                break;
            case 2:
                type = HouseType.Townhouse;
                break;
            case 3:
                type = HouseType.Condominium;
                break;
            default:
                type = HouseType.Unknown;
                break;
        }
        Console.Write("Does the house have an indoor garage (1=Yes/0=No)? ");
        var answer = int.Parse(Console.ReadLine());
        if (answer == 1)
            garage = "Yes";
        else
            garage = "No";
        Console.WriteLine("\nDesired House Type: {0}", type);
        Console.WriteLine("Has indoor garage? {0}", garage);
        return 0;
    }
}
```

نمونه ای از اجرای برنامه

```
Enter the type of house you want to purchase
1. Single Family
```

```
2. Townhouse
3. Condominium
You Choice? 8
Does the house have an indoor garage (1=Yes/0=No)? 2

Desired House Type: Unknown
Has indoor garage?  No
Press any key to continue...
```

گذشته از مقدار نوع int، می توان دیگر نوع های اعداد صحیح را در دستور switch به کار برد. به عنوان مثال، می توان با استفاده از حروف، case ها را اعتبارسنجی کرد. به مثال زیر توجه کنید.

```
using System;
public enum HouseType
{
  Unknown,
  SingleFamily,
  Townhouse,
  Condominium
}
public class Exercise
{
  public static int Main()
  {
    var type = HouseType.Unknown;
    var choice = 0;
    var garage = "";
    Console.WriteLine("Enter the type of house you want to purchase");
    Console.WriteLine("1. Single Family");
    Console.WriteLine("2. Townhouse");
    Console.WriteLine("3. Condominium");
    Console.Write("You Choice? ");
    choice = int.Parse(Console.ReadLine());
    switch (choice)
    {
      case 1:
        type = HouseType.SingleFamily;
        break;
      case 2:
        type = HouseType.Townhouse;
        break;
      case 3:
        type = HouseType.Condominium;
        break;
      default:
```

```
            type = HouseType.Unknown;
            break;
        }
        Console.Write("Does the house have an indoor garage (y/n)? ");
        var answer = char.Parse(Console.ReadLine());
        switch (answer)
        {
          case 'y':
            garage = "Yes";
            break;
          case 'Y':
            garage = "Yes";
            break;
          case 'n':
            garage = "No";
            break;
          case 'N':
            garage = "No";
            break;
          default:
            garage = "Not Specified";
            break;
        }
        Console.WriteLine("\nDesired House Type: {0}", type);
        Console.WriteLine("Has indoor garage? {0}", garage);
        return 0;
    }
}
```

نمونه ای از اجرای برنامه

```
Enter the type of house you want to purchase
1. Single Family
2. Townhouse
3. Condominium
You Choice? 3
Does the house have an indoor garage (y/n)? y

Desired House Type: Condominium
Has indoor garage? Yes
Press any key to continue...
```

دستورات شرطی switch

۱. ابتدا Microsoft Visual Studio را اجرا کنید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

۲. برای ایجاد برنامه ی جدید، در فهرست گزینه ی اصلی، روی File -> New Project... کلیک کنید.

۳. در لیست میانی، گزینه ی Empty Project را انتخاب کنید.

۴. اسم پروژه را به NationalBank3 تغییر دهید و کلید Enter را بزنید.

۵. به منظور ایجاد فایل جدید، به فهرست گزینه ی اصلی مراجعه کرده، Project -> Add New Item... کلیک کنید.

۶. گزینه ی Code File را از لیست میانی انتخاب کنید.

۷. اسم فایل مورد نظر را Customer انتخاب کنید.

۸. حال Add را کلیک کنید.

۹. فایل Customer.cs را به صورت زیر اصلاح کنید.

```
public enum AccountType { Checking, Saving, Other }
public class Customer
{
  public string AccountNumber;
  public AccountType Type;
  public string FullName;
  public double Balance;
  public short PIN;
  public Customer(string acnt = "000-000000-000",AccountType category = AccountType.Other,string name = "John Doe")
  {
    AccountNumber = acnt;
    Type = category;
    FullName = name;
    PIN = 0;
    Balance = 0.00D;
  }
}
```

۱۰. برای ایجاد فایل جدید، در پنجره ی Solution Explorer روی NationalBank3 راست کلیک کرده سپس : NationalBank3 -> Add -> New Item...

۱۱. گزینه ی Code File در لیست میانی باید حتماً انتخاب شده باشد. اسم آن را به Management تغییر دهید و کلید Enter را بزنید.

۱۲. فایل مورد نظر را به ترتیب زیر تکمیل کنید.

```
using System;
  public class Management
  {
    private Customer CreateNewAccount()
    {
      byte typeOfAccount = 0;
```

```
    Customer client = new Customer();
    Console.WriteLine("==========================================");
    Console.WriteLine("==-= National Bank =-=====================");
    Console.WriteLine("------------------------------------------");
    Console.Write("Enter a number for the new account(000-000000-000): ");
    client.AccountNumber = Console.ReadLine();
    Console.WriteLine("What type of account the customer wants to open");
    Console.WriteLine("1 - Checking Account");
    Console.WriteLine("2 - Savings Account");
    Console.Write("Enter account type: ");
    typeOfAccount = byte.Parse(Console.ReadLine());
    if (typeOfAccount == 1)
      client.Type = AccountType.Checking;
    else if (typeOfAccount == 2)
      client.Type = AccountType.Saving;
    else
      client.Type = AccountType.Other;
    Console.Write("Enter customer name: ");
    client.FullName = Console.ReadLine();
    Console.Write("Ask the customer to enter a PIN: ");
    client.PIN = short.Parse(Console.ReadLine());
    return client;
}
public double GetMoney()
{
    double amount = 0;
    Console.Write("Amount: ");
    amount = double.Parse(Console.ReadLine());
    return amount;
}
private void ShowAccountInformation(Customer cust)
{
    Console.WriteLine("==========================================");
    Console.WriteLine("==-= National Bank =-=====================");
    Console.WriteLine("Customer Account Information");
    Console.WriteLine("------------------------------------------");
    Console.WriteLine("Account #:  {0}", cust.AccountNumber);
    Console.WriteLine("Account Type: {0}", cust.Type);
    Console.WriteLine("Full Name:  {0}", cust.FullName);
    Console.WriteLine("PIN #:    {0}", cust.PIN);
    Console.WriteLine("Balance:   {0:F}", cust.Balance);
    Console.WriteLine("==========================================");
}
public static int Main()
{
    double amount = 0;
    byte nextAction = 0;
    Customer accountHolder = null;
    Management registration = new Management();
    Console.Title = "National Bank";
```

```
    accountHolder = registration.CreateNewAccount();
    Console.WriteLine("Enter the customer's initial deposit");
    accountHolder.Balance = registration.GetMoney();
    Console.Clear();
    registration.ShowAccountInformation(accountHolder);
    Console.WriteLine("What do you want to do now?");
    Console.WriteLine("1 - Check account balance");
    Console.WriteLine("2 - Make a deposit");
    Console.WriteLine("3 - Withdraw money");
    Console.WriteLine("4 - Transfer money from one account to another");
    nextAction = byte.Parse(Console.ReadLine());
    switch (nextAction)
    {
      case 1:
        break;
      case 2:
        Console.Write("Enter the Deposit ");
        amount = double.Parse(Console.ReadLine());
        accountHolder.Balance = accountHolder.Balance + amount;
        break;
      case 3:
        Console.Write("Enter the Withdrawal ");
        amount = double.Parse(Console.ReadLine());
        accountHolder.Balance = accountHolder.Balance - amount;
        break;
      case 4:
        Console.WriteLine("Operation not available: You have only one account with us");
        break;
    }
    Console.Clear();
    registration.ShowAccountInformation(accountHolder);
    Console.ReadKey();
    return 0;
  }
}
```

۱۳. کلید F5 را بزنید تا برنامه اجرا شود.

۱۴. اکنون اطلاعات مورد نیاز را به ترتیب زیر وارد کنید.

Account #	202-410443-240
Account Type:	1
Customer Name:	Paul Martin Eloundou

PIN:	8402
Initial Deposit:	750

```
==========================================
==-= National Bank =-=====================
------------------------------------------
Enter a number for the new account(000-000000-000)202-410443-240
What type of account the customer wants to open
1 - Checking Account
2 - Savings Account
Enter account type: 1
Enter customer name: Paul Martin Eloundou
Ask the customer to enter a PIN: 8402
Enter the customer's initial deposit
Amount: 750
```

۱۵. Enter را بزنید.

```
==========================================
==-= National Bank =-=====================
Customer Account Information
------------------------------------------
Account #:  202-410443-240
Account Type: Checking
Full Name:  Paul Martin Eloundou
PIN #:  8402
Balance:  750.00
==========================================
What do you want to do now?
1 - Check account balance
2 - Make a deposit
3 - Withdraw money
4 - Transfer money from one account to another
```

۱۶. در مرحله ی بعدی 2 را تایپ کرده، سپس Enter را فشار دهید.

۱۷. مبلغ را رقم 226.85 وارد کنید و باز Enter را بزنید.

۱۸. از پنجره ی DOS خارج گشته و به محیط برنامه نویسی بازگردید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

Case های ترکیبی

هر یک از case هایی که تاکنون به کار برده ایم، تنها یک احتمال را پیش از اجرای دستور مربوطه درنظر گرفته و بررسی می کرد. همچنین می توان برای اجرای دستور مرتبط (همان دستور) چندین case را با هم ترکیب کرد. برای نیل به این هدف، ابتدا case، سپس مقدار و نقطه ویرگول را وارد کنید. حال، با استفاده از فرمول نام برده case دیگری تایپ کنید. پس از ایجاد case های مورد نظر، می توان دستور دلخواه را اجرا کرد.

مثال

```
using System;
public enum HouseType
{
  Unknown,
  SingleFamily,
  Townhouse,
  Condominium
}
public class Exercise
{
  public static int Main()
  {
    var type = HouseType.Unknown;
    var choice = 0;
    var garage = "";
    Console.WriteLine("Enter the type of house you want to purchase");
    Console.WriteLine("1. Single Family");
    Console.WriteLine("2. Townhouse");
    Console.WriteLine("3. Condominium");
    Console.Write("You Choice? ");
    choice = int.Parse(Console.ReadLine());
    switch (choice)
    {
      case 1:
        type = HouseType.SingleFamily;
        break;
      case 2:
        type = HouseType.Townhouse;
        break;
      case 3:
        type = HouseType.Condominium;
        break;
      default:
        type = HouseType.Unknown;
        break;
    }
```

```
    Console.Write("Does the house have an indoor garage (y/n)? ");
    var answer = char.Parse(Console.ReadLine());
    switch (answer)
    {
      case 'y':
      case 'Y':
        garage = "Yes";
        break;
      case 'n':
      case 'N':
        garage = "No";
        break;
      default:
        garage = "Not Specified";
        break;
    }
    Console.WriteLine("\nDesired House Type: {0}", type);
    Console.WriteLine("Has indoor garage? {0}", garage);
    return 0;
  }
}
```

نتیجه ی زیر به دست می آید.

```
Enter the type of house you want to purchase
1. Single Family
2. Townhouse
3. Condominium
You Choice? 3
Does the house have an indoor garage (y/n)? Y
Desired House Type: Condominium
Has indoor garage? Yes
Press any key to continue...
```

استفاده از Enumeration

یکی از اصلی ترین کاربردهای enumeration، پردازش آن در دستور switch می باشد. برای این منظور، باید مقدار enumeration را به switch ارسال کرد. مقادیر enumeration سپس در دستورات case پردازش می شوند.

مثال

```
using System;
public enum HouseType
{
  Unknown,
```

```
  SingleFamily,
  Townhouse,
  Condominium
}
public class Exercise
{
  public static int Main()
  {
    var PropertyType = "";
    var choice = 0;
    var garage = "";
    Console.WriteLine("Enter the type of house you want to purchase");
    Console.WriteLine("1. Single Family");
    Console.WriteLine("2. Townhouse");
    Console.WriteLine("3. Condominium");
    Console.Write("You Choice? ");
    choice = int.Parse(Console.ReadLine());
    switch ((HouseType)choice)
    {
      case HouseType.SingleFamily:
        PropertyType = "Single Family";
        break;
      case HouseType.Townhouse:
        PropertyType = "Townhouse";
        break;
      case HouseType.Condominium:
        PropertyType = "Condominium";
        break;
      default:
        PropertyType = "Unknown";
        break;
    }
    Console.Write("Does the house have an indoor garage (y/n)? ");
    var answer = char.Parse(Console.ReadLine());
    switch (answer)
    {
      case 'y':
      case 'Y':
        garage = "Yes";
        break;
      case 'n':
      case 'N':
        garage = "No";
        break;
      default:
        garage = "Not Specified";
        break;
    }
    Console.WriteLine("\nDesired House Type: {0}", PropertyType);
    Console.WriteLine("Has indoor garage? {0}", garage);
```

```
    return 0;
  }
}
```

نمونه ای از اجرای برنامه

```
Enter the type of house you want to purchase
1. Single Family
2. Townhouse
3. Condominium
You Choice? 1
Does the house have an indoor garage (y/n)? N
Desired House Type: Single Family
Has indoor garage?  No
Press any key to continue...
```

عطف منطقی AND

مقدمه

تصور کنید دلال املاک و مستغلاتی که از برنامه ی شما استفاده می کند با یک خریدار بالقوه ملاقات دارد و سوالات زیر را از برنامه می پرسد.

```
using System;
public enum HouseType
{
  Unknown,
  SingleFamily,
  Townhouse,
  Condominium
}
public class Program
{
  static int Main()
  {
    var type = HouseType.Unknown;
    var choice = 0;
    var value = 0D;
    Console.WriteLine("Enter the type of house you want to purchase");
    Console.WriteLine("1. Single Family");
    Console.WriteLine("2. Townhouse");
    Console.WriteLine("3. Condominium");
    Console.Write("You Choice? ");
    choice = int.Parse(Console.ReadLine());
    if (choice == 1)
      type = HouseType.SingleFamily;
    if (choice == 2)
```

```
        type = HouseType.Townhouse;
      if (choice == 3)
        type = HouseType.Condominium;
      Console.Write("Up to how much can you afford? $");
      value = double.Parse(Console.ReadLine());
      Console.WriteLine("\nDesired House Type:    {0}", type);
      Console.WriteLine("Maximum value afforded: {0:C}\n", value);
      return 0;
    }
  }
```

تصور کنید مشتری به این سوالات پاسخ می دهد : می گوید که خانه ی تک خانوار (single family house) می خواهد ولی توان پرداخت بیش از $550،000 را ندارد.

```
Enter the type of house you want to purchase
1. Single Family
2. Townhouse
3. Condominium
You Choice? 1
Up to how much can you afford? $550000
Desired House Type:    SingleFamily
Maximum value afforded: $550,000.00
Press any key to continue...
```

هنگام بررسی خانه برای چنین مشتریی، دو پارامتر باید در نظر گرفته و اعتبار سنجی شود : خانه ی مذکور باید برای تک خانوار تعبیه شده باشد و قیمت آن زیر مبلغ $550،001 باشد. بر این اساس، دو دستور به شکل زیر می نویسیم.

```
a. The house is single family
b. The house costs less than $550,000
```

از لیست املاک و مستغلات، چنانچه خانه ای با این مشخصه (تک خانوار بودن آن) پیدا کریدم آن را داخل فهرست املاک مورد نظرمان قرار می دهیم.

Type of House	House
The house is single family	True

از سوی دیگر، در صورت پیدا کردن خانه ای با قیمت $550،000 یا پایین تر از آن، خانه را بررسی می کنیم.

Price Range	Value
$550،000	True

یکی از راه های ترکیب دو مقایسه، پیوند دادن آن ها است. برای مشتری مزبور خانه ای را انتخاب می کنیم که (هر) دو مشخصه ی فوق را دارا باشد. اگر خانه ی مورد نظر town house باشد، طبق خواسته ی مشتری، مقدار شرطی آن غلط (false) خواهد بود. اگر قیمت خانه بیش از $550،000 باشد، باز هم مقدار Boolean Value ناصحیح خواهد بود.

Type of House	House Value	Result
Town House	$625،000	Town House AND $625،000
False	False	False

در C#، عملگر بولی AND با && اجرا می شود.

مثال

```
 using System;
public enum HouseType
{
   Unknown,
   SingleFamily,
   Townhouse,
   Condominium
}
public class Exercise
{
   static int Main()
   {
```

```
        var type = HouseType.Unknown;
        var choice = 0;
        var value = 0D;
        Console.WriteLine("Enter the type of house you want to purchase");
        Console.WriteLine("1. Single Family");
        Console.WriteLine("2. Townhouse");
        Console.WriteLine("3. Condominium");
        Console.Write("You Choice? ");
        choice = int.Parse(Console.ReadLine());
        Console.Write("Up to how much can you afford? $");
        value = double.Parse(Console.ReadLine());
        if (choice == 1)
          type = HouseType.SingleFamily;
        if (choice == 2)
          type = HouseType.Townhouse;
        if (choice == 3)
          type = HouseType.Condominium;
        Console.WriteLine("\nDesired House Type:    {0}", type);
        Console.WriteLine("Maximum value afforded: {0:C}", value);
        if (type == HouseType.SingleFamily && value <= 550000)
          Console.WriteLine("\nDesired House Matched");
        return 0;
    }
}
```

نمونه ای از اجرای برنامه

```
Enter the type of house you want to purchase
1. Single Family
2. Townhouse
3. Condominium
You Choice? 1
Up to how much can you afford? $450000
Desired House Type:    SingleFamily
Maximum value afforded: $450,000.00
Desired House Matched
Press any key to continue...
```

اساساً، ترکیب عطفی منطقی (logical conjunction) دو شرط را با هم ادغام می کند. به منظور بهبود خوانایی برنامه، می توان هر طرف شرط را داخل پرانتز جای داد.

مثال

```
using System;
public enum HouseType
{
   Unknown,
   SingleFamily,
   Townhouse,
   Condominium
}
public class Exercise
{
   static int Main()
   {
      var type = HouseType.Unknown;
      var choice = 0;
      var value = 0D;
        . . .
      if ((type == HouseType.SingleFamily) && (value <= 550000))
         Console.WriteLine("\nDesired House Matched");
      return 0;
   }
}
```

تصور کنید خانه ای با مشخصه ی اول (single family) پیدا کردیم. پس اولین شرط لازم برای مشتری برآورده شد. به کمک عملگر بولی AND، به سراغ معیار دوم می رویم. تصور کنید خانه ای که در نظر گرفته ایم، 500،$750 است : مبلغ مورد نیاز از قدرت خرید مشتری فراتر است. بنابراین، شرط دوم برآورده نمی شود (false است). در جبر بولی AND، اگر هم شرط اول درست باشد، در صورت ناصحیح بودن شرط یا معیار دوم کل شرط غلط تلقی می گردد.

Type of House	House Value	Result
Single Family	$750،500	Single Family AND $750،500
True	False	False

مثال

```
using System;
public enum HouseType
{
```

```
    Unknown,
    SingleFamily,
    Townhouse,
    Condominium
}
class Program
{
    static void Main()
    {
        var type = HouseType.Unknown;
        int choice;
        var value = 0M;
        Console.WriteLine("Enter the type of house you want to purchase");
        Console.WriteLine("1. Single Family");
        Console.WriteLine("2. Townhouse");
        Console.WriteLine("3. Condominium");
        Console.Write("You Choice? ");
        choice = int.Parse(Console.ReadLine());
        if (choice == 1)
            type = HouseType.SingleFamily;
        if (choice == 2)
            type = HouseType.Townhouse;
        if (choice == 3)
            type = HouseType.Condominium;
        Console.Write("Up to how much can you afford? $");
        value = decimal.Parse(Console.ReadLine());
        Console.WriteLine("\nDesired House Type:    {0}", type);
        Console.WriteLine("Maximum value afforded: {0:C}", value);
        if (type == HouseType.SingleFamily && value <= 550000)
            Console.WriteLine("\nDesired House Matched");
        else
            Console.WriteLine("\nThe House Doesn't Match the Desired Criteria");
    }
}
```

نمونه ای از اجرای برنامه

```
Enter the type of house you want to purchase
1. Single Family
2. Townhouse
3. Condominium
You Choice? 1
Up to how much can you afford? $750000
Desired House Type:    SingleFamily
Maximum value afforded: $750,000.00
The House Doesn't Match the Desired Criteria
Press any key to continue...
```

حال تصور کنید خانه ی از نوع townhouse (برای چندین خانوار) پیدا کرده ایم که 000،$420 هزینه بر می دارد. اگرچه دومین شرط صحیح است، اولین معیار در نظر گرفته نشده. در جبر بولی، چنانچه حتی یکی از شرایط ناصحیح باشد، عملیات AND کلاً غلط می شود.

Type of House	House Value	Result
Town House	$420،000	Town House AND $420،000
False	True	False

مثالی از اجرای برنامه، زیر قابل مشاهده می باشد :

```
Enter the type of house you want to purchase
1. Single Family
2. Townhouse
3. Condominium
You Choice? 2
Up to how much can you afford? $420000
Desired House Type:     Townhouse
Maximum value afforded: $420،000.00
The House Doesn't Match the Desired Criteria
Press any key to continue...
```

چنانچه خانه ای تک خانوار (single family) با قیمت $345,000،پیدا کنیم، هر دو شرط برآورده می شود. در جبر بولی، در صورتی که هر دو شرط صحیح باشند، عملیات AND نیز true محسوب می شود.

Type of House	House Value	Result
Single Family	$345،000	Single Family AND $345،000
True	True	True

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

نمونه ای از اجرای برنامه ی بالا را زیر رویت می کنید.

```
Enter the type of house you want to purchase
 1. Single Family
 2. Townhouse
 3. Condominium
 You Choice? 1
 Up to how much can you afford? $345000
 Desired House Type:     SingleFamily
 Maximum value afforded: $345,000.00
 Desired House Matched
 Press any key to continue...
```

چهار جدول بالا را می توان به صورت زیر ادامه داد.

If Condition1 is	If Condition2 is	Condition1 AND Condition2
False	False	False
False	True	False
True	False	False
True	True	True

همان طور که مشاهده می کنید عطف منطقی تنها زمانی به true ارزیابی می شود که هر دو شرط درست باشند. در برخی موارد لازم است چندین شرط (بیش از دو شرط) را با هم ادغام یا ترکیب کنیم. تصور کنید مراجعه کننده ای قصد خرید خانه ای با گنجایش ظرفیت یک خانوار (single family house) و مجهز به پارکینگ (indoor garage) که 000،$450 قیمت آن است را دارد. به عبارت دیگر خانه ی مذکور باید سه شرط زیر را داشته باشد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

a. The house is a single family home

b. The house costs less than $450،001

c. The house has an indoor garage

برنامه ی زیر شرایط بالا را مورد بررسی قرار می دهد.

```
using System;
public enum HouseType
{
   Unknown,
   SingleFamily,
   Townhouse,
   Condominium
}
public class Exercise
{
   static int Main()
   {
      var type = HouseType.Unknown;
      var choice = 0;
      var value = 0D;
      var hasIndoorGarage = false;
      Console.WriteLine("Enter the type of house you want to purchase");
      Console.WriteLine("1. Single Family");
      Console.WriteLine("2. Townhouse");
      Console.WriteLine("3. Condominium");
      Console.Write("You Choice? ");
      choice = int.Parse(Console.ReadLine());
      if (choice == 1)
         type = HouseType.SingleFamily;
      if (choice == 2)
         type = HouseType.Townhouse;
      if (choice == 3)
         type = HouseType.Condominium;
      Console.Write("Up to how much can you afford? $");
      value = double.Parse(Console.ReadLine());
      Console.Write("Does the house have an indoor garage (1=Yes/0=No)? ");
      int ans = int.Parse(Console.ReadLine());
      Console.WriteLine("\nDesired House Type:    {0}", type);
      Console.WriteLine("Maximum value afforded: {0:C}", value);
      Console.Write("House has indoor garage: ");
      if (ans == 1)
         Console.WriteLine("Yes");
      else
         Console.WriteLine("No");
      if ((type == HouseType.SingleFamily) && (value <= 550000) && (ans == 1))
         Console.WriteLine("\nDesired House Matched");
```

```
    else
      Console.WriteLine("\nThe House Doesn't Match the Desired Criteria");
    return 0;
  }
}
```

همان طور که پیش تر ذکر شد، زمانی که دو شرط با هم ترکیب می شوند، compiler ابتدا شرط اول را چک می کند سپس شرط دوم را مورد بررسی قرار می دهد. به همین ترتیب، چنانچه لازم است سه شرط لحاظ شود، compiler در وهله ی اول درستی اولین شرط را ارزیابی می کند.

Type of House
A
Town House
False

در صورتی که شرط اول (یا هر یک از شروط) غلط باشد، کل شرط نادرست اطلاق می گردد ، صرفنظر از نتیجه ی شرط دوم. اگر شرط اول صحیح بود، شرط دوم ارزیابی می شود.

Type of House	Property Value
A	B
Single Family	$655،000
True	False

چنانچه شرط دوم نادرست بود، کل ترکیب غلط تلقی می گردد.

A	B	A && B
True	False	False

حین ارزیابی سه شرط بالا اگر شرط اول یا دوم ناصحیح باشد، به دلیل اینکه کل شرط نادرست محسوب می شود، دیگر دلیلی برای بررسی شرط سوم وجود ندارد. اگر هر دو شرط اول و دوم غلط باشند باز هم ارزیابی شرط سوم لزومی ندارد. فقط در صورت صحیح بودن دو شرط مذکور شرط سوم ارزیابی می شود.

Type of House	Property Value	Indoor Garage
A	B	C
Single Family	$425،650	None
True	True	False

ترکیبی از سه شرط بالا در عطف منطقی بدین صورت نمایش داده می شود : A && B && C. بار دیگر، چنانچه شرط سوم نادرست باشد، کل شرط غلط می شود.

A	B	A && B	C	A && B && C
True	True	True	False	False

مثال

```
Enter the type of house you want to purchase
1. Single Family
2. Townhouse
3. Condominium
You Choice? 1
Up to how much can you afford? $425000
Does the house have an indoor garage (1=Yes/0=No)? 0
```

Desired House Type： SingleFamily
Maximum value afforded：$425،000.00
House has indoor garage：No
The House Doesn't Match the Desired Criteria
Press any key to continue…

A	B	C	A && B && C
False	Don't Care	Don't Care	False
True	False	Don't Care	False
True	True	False	False

کل شرط تنها زمانی صحیح (true) در نظر گرفته می شود که هر سه شرط درست باشند.

A	B	C	A && B && C
False	False	False	False
False	False	True	False
True	False	False	False
True	False	True	False
False	True	False	False
False	True	True	False
True	True	False	False
True	True	True	True

فصل منطقی : or

مقدمه

حال در نظر بگیرید بنگاه معملات املاک هر سه نوع خانه ی single family، townhouse و condominium را دارد. کلیه ی condominium ها فقط یک طبقه دارند. برخی از خانه های single family یک طبقه، برخی دو طبقه و تعدادی دارای سه طبقه می باشند. ولی تمامی خانه های townhouse سه طبقه ای می باشند.

مراجعه کننده ی دیگری قصد خرید خانه را دارد و وی condominium می پسندد. حال چنانچه شرکت ما چه خانه داشته باشد چه condo، نباید بیشتر از یک طبقه باشد (مشتری به هر دلیلی تمایل به بالا رفتن از پله ها را ندارد). هنگام بررسی املاک بنگاه، با در نظر گرفتن شرط بالا، دستور های زیر را می نویسیم.

a. The property is a condominium

b. The property has one story

در صورت پیدا کردن condo، به این خاطر که کلیه ی condo ها تنها یک طبقه دارند، معیار موردنظر برآورده می شود.

Type of House	House
Condominium	True

املاک دیگر، به خصوص آن دسته ای که بیش از یک طبقه دارند، نادیده گرفته می شوند.

Number of Stories	Value
3	False

عملگر شرطی or (||) در صورت ارزیابی عملوند اول به صحیح، دیگر عملوند دوم را ارزیابی نمی کند. چنانچه عملوند اول به false ارزیابی شود، این عملوند دوم است که تعیین یا مشخص می کند که کل عبارت or به صحیح یا غلط ارزیابی شود. ترتیب عملیات or در قالب جدول زیر به نمایش گذاشته شده است.

Condominium	One Story	Condominium OR 1 Story
True	False	True

مثال

```
using System;
public enum HouseType
{
   Unknown,
   SingleFamily,
   Townhouse,
   Condominium
}
public class Exercise
{
   static int Main()
   {
      var type = HouseType.Unknown;
      var choice = 0;
      var stories = 1;
      Console.WriteLine("Enter the type of house you want to purchase");
      Console.WriteLine("1. Single Family");
      Console.WriteLine("2. Townhouse");
      Console.WriteLine("3. Condominium");
      Console.Write("You Choice? ");
      choice = int.Parse(Console.ReadLine());
      if (choice == 1)
         type = HouseType.SingleFamily;
      if (choice == 2)
         type = HouseType.Townhouse;
      if (choice == 3)
         type = HouseType.Condominium;
      Console.Write("How many stories? ");
      stories = int.Parse(Console.ReadLine());
      Console.WriteLine("\nDesired House Type: {0}", type);
      Console.WriteLine("Number of Stories: {0}", stories);
      if ((type == HouseType.Condominium) || (stories == 1))
         Console.WriteLine("\nDesired House Matched");
      else
         Console.WriteLine("\nThe House Doesn't Match the Desired Criteria");
      return 0;
   }
}
```

نمونه ای از اجرای برنامه

Enter the type of house you want to purchase

1. Single Family

2. Townhouse

3. Condominium

You Choice? 3

How many stories? 6

Desired House Type: Condominium

Number of Stories: 6

Desired House Matched

Press any key to **continue**…

اکنون تصور کنید بین املاک موجود در بنگاه، هیچ condominium وجود ندارد. در این صورت، دیگر املاک را در نظر می گیریم.

Type of House	House
Single Family	False

چنانچه تعداد محدودی گزینه ی single family موجود است، به دنبال آن خانه ای می گردیم که فقط یک طبقه داشته باشد. در صورت پیدا کردن گزینه ی مورد نظر، شرط دوم برآورده می شود.

Type of House	One Story	Condominium OR 1 Story
False	True	True

مثال

Enter the type of house you want to purchase

1. Single Family

2. Townhouse

3. Condominium

You Choice? 1

How many stories? 1

Desired House Type: SingleFamily

```
Number of Stories: 1
Desired House Matched
Press any key to continue...
```

اگر گزینه ی condominium را در املاک موجود پیدا کنیم که یک طبقه داشته باشد، هر دو شرط لازم برآورده می شود. جدول زیر این عملیات را به نمایش می گذارد.

Type of House	One Story	Condominium OR 1 Story
False	True	True
True	True	True

برنامه ی زیر مثال بالا را نمایش می دهد.

```
Enter the type of house you want to purchase
1. Single Family
2. Townhouse
3. Condominium
You Choice? 3
How many stories? 1
Desired House Type: Condominium
Number of Stories: 1
Desired House Matched
Press any key to continue...
```

عملیات بولی or تنها زمانی کلاً نتیجه ی false تولید می کند که هر دو شرط غلط باشند.

If Condition1 is	If Condition2 is	Condition1 OR Condition2
False	True	True
True	False	True
True	True	True

False	False	False

مثالی دیگر

```
Enter the type of house you want to purchase
1. Single Family
2. Townhouse
3. Condominium
You Choice? 2
How many stories? 2
Desired House Type: Townhouse
Number of Stories: 2
The House Doesn't Match the Desired Criteria
Press any key to continue...
```

فصل های ترکیبی

گاهی اوقات لازم است به جای دوشرط، سه شرط ارزیابی شود. برای این منظور از فصل های ترکیبی کمک گرفته می شود.

نحوه ی شمارش در حلقه

تکرار/ حلقه ی شرطی

حلقه یک نوع دستور شرطی است که به بررسی شرط و اجرای دستور مورد نظر ادامه می دهد تا شرط بیان شده false ارزیابی شود.

معرفی تکرار شرطی

۱. Microsoft Visual Studio را اجرا کنید.

۲. برای ایجاد برنامه ی جدید، در فهرست گزینه ی اصلی، روی File -> New Project... کلیک کنید.

۳. گزینه ی Empty Project را از لیست میانی انتخاب کنید.

۴. اسم پروژه را NationalBank4 انتخاب کنید و کلید Enter را بزنید.

۵. به منظور ایجاد فایل جدید، به فهرست گزینه ی اصلی مراجعه کرده، سپس Project -> Add New Item... .

۶. روی Code File، واقع در لیست میانی کلیک کنید.

۷. اسم فایل مورد نظر را به Customer تغییر دهید.

۸. حال Add را کلیک کنید.

۹. فایل Customer.cs را به ترتیب زیر اصلاح کنید.

```
public enum AccountType { Checking, Saving, Other }
public class Customer
{
  public string AccountNumber;
  public AccountType Type;
  public string FullName;
  public double Balance;
  public short PIN;
  public Customer(string acnt = "000-000000-000", AccountType category = AccountType.Other,string name = "John Doe")
  {
    AccountNumber = acnt;
    Type = category;
    FullName = name;
    PIN = 0;
    Balance = 0.00D;
  }
}
```

۱۰. برای ایجاد فایل جدید، در پنجره ی Solution Explorer، راست کلیک کرده سپس NationalBank4 -> Add -> New Item... .

۱۱. پس از انتخاب Code File از لیست میانی اسم مربوطه را به Management تغییر داده و کلید Enter را بزنید.

۱۲. فایل را به صورت زیر تکمیل کنید.

```
using System;
public class Management
{
  private Customer CreateNewAccount()
  {
    byte typeOfAccount = 0;
    Customer client = new Customer();
    Console.WriteLine("=============================================");
    Console.WriteLine("==-= National Bank =-=====================");
    Console.WriteLine("---------------------------------------");
    Console.Write("Enter a number for the new account(000-000000-000): ");
    client.AccountNumber = Console.ReadLine();
    Console.WriteLine("What type of account the customer wants to open");
    Console.WriteLine("1 - Checking Account");
    Console.WriteLine("2 - Savings Account");
    Console.Write("Enter account type: ");
```

```
    typeOfAccount = byte.Parse(Console.ReadLine());
    if (typeOfAccount == 1)
      client.Type = AccountType.Checking;
    else if (typeOfAccount == 2)
      client.Type = AccountType.Saving;
    else
      client.Type = AccountType.Other;
    Console.Write("Enter customer name: ");
    client.FullName = Console.ReadLine();
    Console.Write("Ask the customer to enter a PIN: ");
    client.PIN = short.Parse(Console.ReadLine());
    return client;
  }
  public double GetMoney()
  {
    double amount = 0;
    Console.Write("Amount: ");
    amount = double.Parse(Console.ReadLine());
    return amount;
  }
  private void ShowAccountInformation(Customer cust)
  {
    Console.WriteLine("=========================================");
    Console.WriteLine("==-= National Bank =-====================");
    Console.WriteLine("Customer Account Information");
    Console.WriteLine("-----------------------------------------");
    Console.WriteLine("Account #:  {0}", cust.AccountNumber);
    Console.WriteLine("Account Type: {0}", cust.Type);
    Console.WriteLine("Full Name:  {0}", cust.FullName);
    Console.WriteLine("PIN #:    {0}", cust.PIN);
    Console.WriteLine("Balance:   {0:F}", cust.Balance);
    Console.WriteLine("=========================================");
  }
  public static int Main()
  {
    return 0;
  }
}
```

حلقه ی While

یکی از عملگرهایی که برای اجرای حلقه به کار می رود while نام دارد. برای ایجاد حلقه ی while از فرمول زیر استفاده کنید.

```
while(Condition) Statement;
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

روی بخش دلخواه راست کلیک کرده، سپس گزینه ی Insert Snippet… را انتخاب کنید. حال دوبار روی لِیبِل Visual C# کلیک کنید. در لیستی که نمایان می شود، روی while دوبار کلیک کنید.

به منظور اجرای شرط while، compiler ابتدا *Condition* را بررسی می کند. در صورت صحیح بودن *Condition*، *Statement* اجرا می شود. پس از اجرای *Statement*، *Condition* بار دیگر بررسی می شود. مادام اینکه *Condition* صحیح / true باشد، *Statement* به صورت تکرار چک می شود. فقط زمانی که *Condition* (به) غلط ارزیابی شود، حلقه اتمام یافته و از آن خارج می شود.

مثال

```
using System;
public class Exercise
{
  public static int Main()
  {
    var stories = 0;
    while (stories <= 4)
    {
      Console.WriteLine("Number {0}", stories);
      stories++;
    }
    return 0;
  }
}
```

نتیجه ی زیر به دست می آید.

```
Number 0
Number 1
Number 2
Number 3
Number 4
Press any key to continue...
```

برای اجرای صحیح شرط while، باید مکانیزمی برای compiler در نظر بگیرید تا بتواند از مقدار ارجاع برای شرط، متغیر یا عبارت مورد نظر استفاده کند (که گاهی در قالب / فرم متغیری است که مقداردهی اولیه می شود، اگرچه ممکن است در قالب عبارت های دیگر نیز ظاهر شود). شرط while به ترتیب زیر با مثال روشن میشود.

به کاربردن while

۱. فایل Management.cs را بدین صورت اصلاح کنید.

```
using System;
public class Management
{
  private int CreateNewAccount()
  {
    . . . No Change
  }
  public static int Main()
  {
    int? Counter = 1;
    Customer accountHolder = null;
    Management registration = new Management();
    Console.Title = "National Bank";
    while (Counter < 5)
    {
      accountHolder = registration.CreateNewAccount();
      Console.WriteLine("Enter the customer's initial deposit");
      accountHolder.Balance = registration.GetMoney();
      Console.Clear();
      registration.ShowAccountInformation(accountHolder);
      Counter++;
      Console.WriteLine("Press Enter for Next Action");
      Console.ReadKey();
      Console.Clear();
    }
    return 0;
  }
}
```

۲. رای اجرای برنامه می توانید کلید F5 را بزنید.

۳. طلاعات مورد نیاز را به ترتیب زیر وارد کنید.

Account #	202-410443-240
Account Type:	1
Customer Name:	Paul Martin Eloundou

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

PIN:	8402
Initial Deposit:	750

Account #:	202-103344-042
Account Type:	2
Customer Name:	Jimmy Simms
PIN:	2468
Initial Deposit	500

Account #	410-240301-443
Account Type:	2
Customer Name:	Eya Miri
PIN:	1119
Initial Deposit:	1500

Account #	718-202404-240
Account Type:	1

Customer Name:	Alain Gassila
PIN:	2226
Initial Deposit:	220.8

٤. اکنون Enter را بزنید.

دستور do...while

با حلقه ی while ابتدا شرط بررسی می شود، سپس دستور اجرا می شود. در صورت غلط بودن شرط، دستور هیچگاه اجرا نمی شود.

مثال

```
using System;
public class Exercise
{
  public static int Main()
  {
    var stories = 5;
    while (stories <= 4)
    {
      Console.WriteLine("Number {0}", stories);
      stories++;
    }
    return 0;
  }
}
```

زمانی که برنامه اجرا می شود هیچ چیز از حلقه ی while اجرا نمی شود، زیرا هنگامی که شرط در ابتدای امر بررسی می شود، به علت غلط بودن آن، compiler هیچگاه به اجرای دستور نمی رسد. در برخی موارد ممکن است لازم شود دستوری را پیش از بررسی شرط آن برای اولین بار، اجرا کنید. برای این منظور دستور do...while را به کار ببرید. فرمول آن به شرح زیر است.

```
do while (Condition);
```

دستور مزبور (do…while) ابتدا Statement را اجرا می کند، به دنبال آن Condition را بررسی می کند. اگر Condition صحیح بود، سپس Statement را دوباره اجرا می کند. تا زمانی که Condition درست باشد، Statement به صورت تکرار اجرا می شود. زمانی که Condition به false ارزیابی شود، تکرار (اجرای مداوم دستور) به پایان می رسد.

چنانچه دستور مورد نظر کوتاه بود، به طور مثال متشکل از تنها یک خط بود، می توان به راحتی آن را پس از do قرار داد. مشابه دستورات if و while، Condition دستور do…while باید داخل پرانتز قرار داده شود. به خاطر داشته باشید که کل دستور do…while باید به علامت نقطه ویرگول " ; " ختم شود.

ورژن دیگری از برنامه ی بالا بدین صورت می باشد.

```
using System;
public class Exercise
{
  public static int Main()
  {
    var stories = 0;
    do
      Console.WriteLine("Number {0}", stories++);
    while (stories <= 4);
    return 0;
  }
}
```

نتیجه ی زیر حاصل می گردد.

```
Number 0
Number 1
Number 2
Number 3
Number 4
Press any key to continue…
```

چنانچه Statement مورد نظر طولانی بوده و بیش از یک خط جا می گیرد، آن را با "{" باز آغاز کرده و با "}" بسته به پایان برسانید.

شمارش و تکرار

۱. فایل Management.cs را به ترتیب زیر اصلاح کنید.

```
using System;
public class Management
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
{
    . . . No Change
  public static int Main()
  {
    double amount = 0;
    byte nextAction = 0;
    Customer accountHolder = null;
    Management registration = new Management();
    Console.Title = "National Bank";
    accountHolder = registration.CreateNewAccount();
    Console.WriteLine("Enter the customer's initial deposit");
    accountHolder.Balance = registration.GetMoney();
    Console.Clear();
    registration.ShowAccountInformation(accountHolder);
    do
    {
      Console.WriteLine("What do you want to do now?");
      Console.WriteLine("1 - Check account balance");
      Console.WriteLine("2 - Make a deposit");
      Console.WriteLine("3 - Withdraw money");
      Console.WriteLine("4 - Transfer money from one account to another");
      nextAction = byte.Parse(Console.ReadLine());
    } while ((nextAction < 1) || (nextAction > 4));
    switch (nextAction)
    {
      case 1:
        break;
      case 2:
        Console.Write("Enter the Deposit ");
        amount = double.Parse(Console.ReadLine());
        accountHolder.Balance = accountHolder.Balance + amount;
        break;
      case 3:
        Console.Write("Enter the Withdrawal ");
        amount = double.Parse(Console.ReadLine());
        accountHolder.Balance = accountHolder.Balance - amount;
        break;
      case 4:
        Console.WriteLine("Operation not available: You have only one account with us");
        break;
    }
    registration.ShowAccountInformation(accountHolder);
    Console.ReadKey();
    return 0;
  }
}
```

۲. F5 را زده تا برنامه اجرا شود.

۳. مقادیر مورد نیاز را بدین صورت وارد کنید.

Account Number:	301-240410-202
Account Type:	1
Customer Name:	Frank Trombs
PIN:	9731
Initial Deposit:	40.1

۴. حال کلید Enter را بزنید.

```
==========================================
==-= National Bank =-=====================
Customer Account Information
------------------------------------------
Account #: 301-240410-202
Account Type: Checking
Full Name: Frank Trombs
PIN #: 9731
Balance: 40.10
==========================================
What do you want to do now?
1 - Check account balance
2 - Make a deposit
3 - Withdraw money
4 - Transfer money from one account to another
Enter your choice:
```

۵. زمانی شما پرسیده شد چه عملیاتی می خواهید انجام دهید، 2 را تایپ کنید تا به حساب پول واریز شود سپس کلید Enter را فشار دهید.

۶. اکنون 125.85 وارد کرده و Enter را بزنید.

```
==========================================
==-= National Bank =-=====================
Customer Account Information
------------------------------------------
Account #: 301-240410-202
Account Type: Checking
Full Name: Frank Trombs
PIN #: 9731
Balance: 165.95
==========================================
```

۷. پس از زدن Enter پنجره ی DOS بسته شده و به محیط برنامه نویسی بازگردانده می شوید.

۸. از محیط برنامه نویسی خارج شوید.

مدیریت دستورات شرطی

For

دستور for اغلب به منظور شمارش تعدادی آیتم مورد استفاده قرار می گیرد. در ساختار معمولَش، حلقه ی for به سه بخش تقسیم می شود. اولین بخش، نقطه ی آغاز شمارش را مشخص می کند. بخش دوم حد شمارش را تعیین می کند. آخرین بخش تکرار یا تناوب شمارش را تعیین می کند. ترکیب نحوی حلقه ی for به این صورت می باشد.

```
for(Start; End; Frequency) Statement;
```

عبارت *Start* متغیری است که به عنوان مقدار اولیه (آغازین) تخصیص داده می شود که می تواند ;Count = 0 باشد (*Start* مقدار آغازین یا نقطه ی شروع حلقه است که می تواند ۰ باشد).

عبارت *End* انتهای فرایند شمارش را تعیین می کند (کجا شمارش به پایان برسد). مثال ;Count < 24. مثال مذکور نشان می دهد که شمارش تا متغیر Count که کوچکتر از 24 باشد ادامه می یابد (*End* نقطه ی پایان حلقه است که مشخص می کند مقدار count تا عددی کوچکتر از ۲۴ ادامه پیدا کند). هنگامی که شمارش به ۲۴ می رسد، به این خاطر که در مثال مورد نظر خود عدد ۲۴ حساب نمی شود، شمارش خاتمه می یابد. برای تعیین حد شمارش، از عملگرهای مقایسه ی >= یا <= استفاده می شود.

عبارت *Frequency* برای compiler مشخص می کند چه مقدار (مقدار عددی که در هر بار تکرار حلقه for به نقطه ابتدایی اضافه و یا کم شود) پیش از ادامه ی حلقه اضافه یا کسر کند. این عبارت می تواند عملیات افزایشی مثل Count++ باشد.

مثال

```
using System;
public class Exercise
{
  public static int Main()
  {
    for (var stories = 0; stories <= 4; stories++)
      Console.WriteLine("Number {0}", stories);
    return 0;
  }
}
```

نتیجه ی زیر به دست می آید.

```
Number 1
Number 2
Number 3
Number 4
Press any key to continue...
```

تودرتو کردن دستور شرطی

برنامه ی زیر را در نظر بگیرید.

```
using System;
public class Exercise
{
  public static int Main()
  {
    var typeOfHome = 0;
    do
    {
      Console.WriteLine("What Type of House Would you Like to Purchase?");
      Console.WriteLine("1 - Single Family");
      Console.WriteLine("2 - Town House");
      Console.WriteLine("3 - Condominium");
      Console.Write("Your Choice? ");
      typeOfHome = int.Parse(Console.ReadLine());
    } while ((typeOfHome < 1) || (typeOfHome > 3));
    if (typeOfHome == 1)
      Console.WriteLine("\nType of Home: Single Family");
    else if (typeOfHome == 2)
      Console.WriteLine("\nType of Home: Town House");
    else if (typeOfHome == 3)
      Console.WriteLine("\nType of Home: Condominium");
    return 0;
  }
}
```

از برنامه ی بالا به منظور درخواست یکی از اعداد ۱،۲ یا ۳ از کاربر استفاده می شود. هر عدد دیگری کمتر از ۱ و بیشتر از ۳ پذیرفته نیست.

مثال

```
What Type of House Would you Like to Purchase?
 1 - Single Family
 2 - Town House
```

```
3 - Condominium
Your Choice? 8
What Type of House Would you Like to Purchase?
1 - Single Family
2 - Town Hou    3 - Condominium
Your Choice? 6
What Type of House Would you Like to Purchase?
1 - Single Family
2 - Town House
3 - Condominium
Your Choice? 3
Type of Home: Condominium
Press any key to continue...
```

اگر کاربر مقداری نامعتبر وارد کند، سوال بار دیگر تکرار می شود. برای حل این مسئله می توان دستور شرطی را داخل دستور شرطی دیگر گنجاند (نوشت) که به آن nesting یا تودرتو کردن می گویند. نحوه ی تودرتو کردن دستور شرطی، زیر با مثال توضیح داده شده است.

```
using System;
public class Exercise
{
  public static int Main()
  {
    var typeOfHome = 0;
    do
    {
      Console.WriteLine("What Type of House Would you Like to Purchase?");
      Console.WriteLine("1 - Single Family");
      Console.WriteLine("2 - Townhouse");
      Console.WriteLine("3 - Condominium");
      Console.Write("Your Choice? ");
      typeOfHome = int.Parse(Console.ReadLine());
      if ((typeOfHome < 1) || (typeOfHome > 3))
        Console.WriteLine("Invalid Choice: Please try again");
    } while ((typeOfHome < 1) || (typeOfHome > 3));
    if (typeOfHome == 1)
      Console.WriteLine("\nType of Home: Single Family");
    else if (typeOfHome == 2)
      Console.WriteLine("\nType of Home: Townhouse");
    else if (typeOfHome == 3)
      Console.WriteLine("\nType of Home: Condominium");
    return 0;
  }
}
```

مثالی دیگر

```
What Type of House Would you Like to Purchase?
1 - Single Family
2 - Town House
3 - Condominium
Your Choice? 0
Invalid Choice: Please try again
What Type of House Would you Like to Purchase?
1 - Single Family
2 - Town House
3 - Condominium
Your Choice? 6
Invalid Choice: Please try againe
What Type of House Would you Like to Purchase?
1 - Single Family
2 - Town House
3 - Condominium
Your Choice? 2
Type of Home: Town House
Press any key to continue...
```

تودرتو کردن شرط ها

۱. فایل Management.csc را به ترتیب زیر اصلاح کنید.

```
using System;
public class Management
{
  private Customer CreateNewAccount()
  {
     . . . No Change
    return client;
  }
  public double GetMoney()
  {
    double amount = 0;
    Console.Write("Amount: ");
    amount = double.Parse(Console.ReadLine());
    return amount;
  }
  private void ShowAccountInformation(Customer cust)
  {
    Console.WriteLine("================================================");
    Console.WriteLine("==-= National Bank =-=======================");
    Console.WriteLine("Customer Account Information");
    Console.WriteLine("------------------------------------------");
    Console.WriteLine("Account #:  {0}", cust.AccountNumber);
```

```
    Console.WriteLine("Account Type: {0}", cust.Type);
    Console.WriteLine("Full Name:  {0}", cust.FullName);
    Console.WriteLine("PIN #:    {0}", cust.PIN);
    Console.WriteLine("Balance:   {0:F}", cust.Balance);
    Console.WriteLine("=========================================");
}
public static int Main()
{
    double amount = 0;
    byte nextAction = 0;
    Customer accountHolder = null;
    Management registration = new Management();
    Console.Title = "National Bank";
    accountHolder = registration.CreateNewAccount();
    Console.WriteLine("Enter the customer's initial deposit");
    accountHolder.Balance = registration.GetMoney();
    Console.Clear();
    registration.ShowAccountInformation(accountHolder);
    do
    {
      Console.WriteLine("What do you want to do now?");
      Console.WriteLine("1 - Check account balance");
      Console.WriteLine("2 - Make a deposit");
      Console.WriteLine("3 - Withdraw money");
      Console.WriteLine("4 - Transfer money from one account to another");
      Console.Write("Enter your choice: ");
      nextAction = byte.Parse(Console.ReadLine());
      Console.Clear();
      switch (nextAction)
      {
        case 1:
          Console.Clear();
          registration.ShowAccountInformation(accountHolder);
          Console.Write("Press Enter for next operation");
          Console.ReadKey();
          break;
        case 2:
          Console.Write("Enter Deposit ");
          amount = registration.GetMoney();
          accountHolder.Balance = accountHolder.Balance + amount;
          Console.Clear();
          registration.ShowAccountInformation(accountHolder);
          break;
        case 3:
          Console.Write("Enter Withdrawal ");
          amount = registration.GetMoney();

          if (amount > accountHolder.Balance)
          {
            Console.WriteLine("You are not allowed to withdraw more money than your account has.");
```

```
            Console.ReadKey();
          }
          else
            accountHolder.Balance = accountHolder.Balance - amount;
          Console.Clear();
          registration.ShowAccountInformation(accountHolder);
          break;
        case 4:
          Console.WriteLine("Operation not available: You have only one account with us");
          break;
      }
      if ((nextAction < 1) || (nextAction > 4))
        Console.WriteLine("Invalid Action: Please enter a value between 1 and 4");
    } while ((nextAction >= 1) && (nextAction <= 4));
    Console.ReadKey();
    return 0;
  }
}
```

۲. با زدن کلید F5 برنامه را اجرا کنید.

۳. اطلاعات لازم را به صورت زیر وارد کنید.

Account #	**202-410443-240**
Account Type:	1
Customer Name:	Paul Martin Eloundou
PIN:	8402
Initial Deposit:	750

۴. کلید Enter را بزنید.

```
==========================================
==-= National Bank =-=====================
Customer Account Information
------------------------------------------
Account #:  202-410443-240
Account Type: Checking
Full Name:  Paul Martin Eloundou
PIN #:  8402
Balance:  750.00
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
==========================================
What do you want to do now?
1 - Check account balance
2 - Make a deposit
3 - Withdraw money
4 - Transfer money from one account to another
Enter your choice:
```

۵. 3 را تایپ کنید تا از حساب پول برداشته شود، سپس کلید Enter را فشار دهید.

۶. 425.85 را به عنوان مبلغ برداشتی تایپ کنید و Enter را بزنید.

۷. در مرحله ی بعد عدد 2 را وارد کرده و بعد کلید Enter را بزنید.

۸. مبلغ 22.84 را تایپ کنید و Enter را بزنید.

```
==========================================
==-= National Bank =-======================
 Customer Account Information
 ----------------------------------------
 Account #:  202-410443-240
 Account Type: Checking
 Full Name:  Paul Martin Eloundou
 PIN #:  8402
Balance:  346.99
 ==========================================
 What do you want to do now?
 1 - Check account balance
 2 - Make a deposit
 3 - Withdraw money
4 - Transfer money from one account to another
 Enter your choice:
```

۹. حال 3 را وارد کنید و Enter را بزنید.

۱۰. مقدار مبلغ برداشتی را 500 وارد کنید و Enter را بزنید.

```
Enter Withdrawal Amount: 500

You are not allowed to withdraw more money than your account has.
```

۱۱. Enter را بزنید .

```
==========================================
=-= National Bank =-======================
ustomer Account Information
----------------------------------------
ccount #:  202-410443-240
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
ccount Type: Checking
ull Name:  Paul Martin Eloundou
IN #:  8402
alance:  346.99
========================================
hat do you want to do now?
 - Check account balance
 - Make a deposit
 - Withdraw money
 - Transfer money from one account to another
nter your choice:
```

۱۲. 0 را تایپ کنید.

۱۳. با زدن Enter از DOS خارج گشته و به محیط برنامه نویسی باز گردید.

قفل کردن تراکنش

زمانی که شما به ایجاد برنامه های (کاربردی) سرویس گیرنده / سرویس دهنده یا منابع فشرده می پردازید، خواهید دانست که امکان دسترسی چندین کاربر یا بیش از یک منبع به برنامه ی شما یا انجام عملیاتی بر شئی از برنامه ی شما وجود دارد. مثال آن تلاش چندین شخص بر افزودن رکورد یا خط جدید به یک فایل یکسان است. در برخی موارد با چنین رخدادی موافق هستید ولی در برخی شرایط سعی دارید از بروز آن جلوگیری کنید.

به منظور جلوگیری از دسترسی کاربر یا یک منبع به شئ از برنامه ی کاربردی مورد نظر، لازم است آن قسمت از تراکنش که عملیات در آن جایز نیست قفل ایجاد شود. برای نیل به این هدف فرمول زیر را به کاربرید.

```
Declare a Variable
lock(Variable)
{
   // Do what you want here
}
// The variable has been released
```

نحوه ی قفل کردن تراکنش

۱. فایل Management.cs را به صورت زیر اصلاح کنید.

```
using System;
   public class Management
   {
```

```
. . . No Change
public static int Main()
{
  double amount = 0;
  byte nextAction = 0;
  Customer accountHolder = null;
  Management registration = new Management();
  Console.Title = "National Bank";
  accountHolder = registration.CreateNewAccount();
  Console.WriteLine("Enter the customer's initial deposit");
  accountHolder.Balance = registration.GetMoney();
  Console.Clear();
  registration.ShowAccountInformation(accountHolder);
  do
  {
    Console.WriteLine("What do you want to do now?");
    Console.WriteLine("1 - Check account balance");
    Console.WriteLine("2 - Make a deposit");
    Console.WriteLine("3 - Withdraw money");
    Console.WriteLine("4 - Transfer money from one account to another");
    Console.Write("Enter your choice: ");
    nextAction = byte.Parse(Console.ReadLine());
    Console.Clear();
    // Don't allow wny other operation on this section
    // while this section is executing
    lock (registration)
    {
      switch (nextAction)
      {
        case 1:
          Console.Clear();
          registration.ShowAccountInformation(accountHolder);
          Console.Write("Press Enter for next operation");
          Console.ReadKey();
          break;
        case 2:
          Console.Write("Enter Deposit ");
          amount = registration.GetMoney();
          accountHolder.Balance = accountHolder.Balance + amount;
          Console.Clear();
          registration.ShowAccountInformation(accountHolder);
          break;
        case 3:
          Console.Write("Enter Withdrawal ");
          amount = registration.GetMoney();
          if (amount > accountHolder.Balance)
          {
            Console.WriteLine(
    "You are not allowed to withdraw more money than your account has.");
            Console.ReadKey();
```

```
            }
            else
              accountHolder.Balance = accountHolder.Balance - amount;
            Console.Clear();
            registration.ShowAccountInformation(accountHolder);
            break;
          case 4:
            Console.WriteLine(
           "Operation not available: You have only one account with us");
            break;
        }
      }
      // Other objects can access this section now
      if ((nextAction < 1) || (nextAction > 4))
        Console.WriteLine("Invalid Action: Please enter a value between 1 and 4");
    } while ((nextAction >= 1) && (nextAction <= 4));
    Console.ReadKey();
    return 0;
  }
}
```

۲. کلید F5 را زده تا برنامه اجرا شود.

۳. کلیه ی مراحل بخش پیشین را دنبال کنید.

۴. به محیط برنامه نویسی خود بازگردید.

ایجاد وقفه در جریان دستور شرطی

دستور break به منظور ایجاد وقفه در یک حلقه یا متوقف کردن آن به کار می رود. فرمول دستور مذکور به شرح زیر می باشد.

```
break;
```

اگرچه break از تنها یک کلمه تشکیل شده است، اما به خودی خود یک دستور کامل تلقی می گردد؛بنابراین، باید همیشه در خط مختص به خودش قرار گیرد (که به افزایش و بهبود خوانایی برنامه کمک شایانی می کند).

دستور break برای نزدیک ترین دستور پیشین (دستوری که بلافاصله break پس از آن در خط بعدی قرار گرفته) به کار بسته می شود، البته منوط به اینکه دستور مذکور (پیشین) قابل اجرا باشد. دستور break در شرط while، do...while و همچنین در حلقه های for به منظور متوقف کردن عمل در حال اجرا بکار می رود. در مثال زیر تعداد طبقات یک ساختمان از ۱ تا ۱۲ شمرده می شود ولی با استفاده از دستور break حین شمارش عدد ۳ برنامه متوقف می گردد.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
using System;
 public class Exercise
 {
   public static int Main()
   {
     for (var stories = 1; stories <= 12; stories++)
     {
       Console.WriteLine("Story {0}", stories);
       if (stories == 3)
         break;
     }
     return 0;
   }
 }
```

نتیجه ی زیر به دست می آید.

```
Story 1
Story 2
Story 3
Press any key to continue...
```

ادامه دادن دستور شرطی

دستور continue از فرمول زیر استفاده می کند.

```
continue;
```

هنگام پردازش حلقه، چنانچه دستور مورد نظر به مقدار نادرستی برخورد کرد، شما می توانید با استفاده از دستور continue در دستورات شرطی while، do…while و for دستور بعدی را نادیده گرفته یا از مقدار غلط و نامعتبر بولی مستقیم به مقدار معتبر بعدی بروید (این برخلاف دستور break می باشد که به طور کلی از حلقه خارج می گردد). مشابه دستور break، کلیدواژه ی continue نزدیک ترین دستور شرطی پیش از خود را تحت تاثیر قرار می دهد و باید در خطی مجزا قرار گیرد. مثال زیر باید طبقات یک ساختمان را از ۱ تا ۶ بشمارد، به نحوه ی استفاده از دستور continue در آن توجه کنید.

```
using System;
 public class Exercise
 {
   public static int Main()
   {
     for (var stories = 1; stories <= 6; stories++)
     {
       if (stories == 3)
```

```
        continue;
      Console.WriteLine("Story {0}", stories);
    }
    return 0;
  }
}
```

نتیجه

```
Story 1
Story 2
Story 4
Story 5
Story 6
Press any key to continue...
```

همان طور که مشاهده می کنید، compiler هنگامی که به عدد ۳ می رسد آن را به طور کامل نادیده گرفته و بقیه ی فرایند را ادامه می دهد.

اصلاح مقداری در حلقه

داخل حلقه، می توان علامتی گذاشت که بر تکامل یک دستور نظارت می کند، از این طریق هنگامی که تگ / دستور به مقدار معینی می رسد به جای نادیده گرفتن تنها یک مقدار (پریدن به اندازه ی یک مقدار/ بجای skip کردن یک مقدار، چندین مقدار جلو برود) به مقدار دلخواه شما می پرد. برای این منظور، در حلقه به مقدار جاری توجه کنید و زمانی که به مقدار مورد نظر رسید، آن را تغییر دهید. در مثال زیر از برنامه خواسته شده از ۱ تا ۱۵ بشمارد.

```
using System;
public class Exercise
{
  static int Main()
  {
    for (var story = 0; story < 15; story++)
    {
      if (story == 6)
        story = 10;
      Console.WriteLine("Elevator at: {0}", story);
    }
    return 0;
  }
}
```

نتیجه

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
Elevator at: 0
Elevator at: 1
Elevator at: 2
Elevator at: 3
Elevator at: 4
Elevator at: 5
Elevator at: 10
Elevator at: 11
Elevator at: 12
Elevator at: 13
Elevator at: 14
Press any key to continue...
```

همان طور که مشاهده می کنید، به محض رسیدن به عدد ۶، حلقه مستقیم به مقدار ۱۰ می پرد.

رفتن به لیبل تعیین شده

دستور goto یکی از دستورات پرش محسوب می شود که کنترل (اجرای برنامه) را به بخش (بخش نام گذاری شده / لیبل تعیین شده) دیگری از تابع همان برنامه انتقال می دهد. برای استفاده از دستور goto باید اسمی را در بخش مورد نظر در تابع درج کرد. لازم به ذکر است که با لیبل (اسم) نشان می دهیم برنامه باید کنترل را به کجا در تابع همان برنامه انتقال دهد. حال، اسم یا لیبل متشکل از تنها یک کلمه است و از همان قوانینی که برای تعیین اسم در C++ تعریف کردیم پیروی می کند. اسم مورد نظر (اسم متواند هر چیزی باشد)، به دنبال آن دونقطه ":" . زیر مثالی را مشاهده می کنید که در آن برنامه طبقات یک ساختمان ۱۴ طبقه ای را می شمارد.

```
using System;
 public class Exercise
{
   public static int Main()
   {
     for (var stories = 1; stories <= 14; stories++)
     {
       if (stories == 4)
         goto CountUpTo3;
       Console.WriteLine("Story {0}", stories);
     }
   CountUpTo3:
     Console.WriteLine("Our homes have only up to 3 levels\n");
     return 0;
   }
 }
```

نتیجه

```
Story 1
Story 2
Story 3
Our homes have only up to 3 levels
Press any key to continue...
```

رفتن به یک لیبل

۱. فایل Management.cs را به ترتیب زیر اصلاح کنید.

```
using System;
  public class Management
  {
     . . . No Change
    public static int Main()
    {
       . . . No Change
      registration.ShowAccountInformation(accountHolder);
      do
      {
         . . . No Change
        lock (registration)
        {
          switch (nextAction)
          {
            case 0:
              goto EndTime;
              break;
            case 1:
              Console.Clear();
              registration.ShowAccountInformation(accountHolder);
              Console.Write("Press Enter for next operation");
              Console.ReadKey();
              break;
             . . . No Change
          }
        }
        if ((nextAction < 1) || (nextAction > 4))
          Console.WriteLine("Invalid Action: Please enter a value between 1 and 4");
      } while ((nextAction >= 1) && (nextAction <= 4));
    EndTime:
      Console.ReadKey();
      return 0;
    }
  }
```

۲. F5 را زده تا برنامه اجرا شود.

۳. اطلاعات مورد نیاز را به صورت زیر وارد کنید.

Account #:	202-103344-042
Account Type:	2
Customer Name:	Jimmy Simms
PIN:	2468
Initial Deposit	500

۴. کلید Enter را فشار دهید.

```
==========================================
==-= National Bank =-=====================
Customer Account Information
------------------------------------------
Account #:  202-103344-042
 Account Type: Saving
 Full Name:  Jimmy Simms
 PIN #:  2468
 Balance:  500.00
 ==========================================
 What do you want to do now?
 1 - Check account balance
 2 - Make a deposit
 3 - Withdraw money
 4 - Transfer money from one account to another
 Enter your choice:
```

۵. عدد 2 را وارد کنید تا سپرده ایجاد شود، سپس Enter را بزنید.

۶. مقدار 328.66 را به عنوان مبلغ واریزی وارد کنید و Enter را بزنید.

```
==========================================
=-= National Bank =-=====================
ustomer Account Information
------------------------------------------
ccount #:  202-103344-042
ccount Type: Saving
ull Name:  Jimmy Simms
IN #:  2468
alance:  828.66
==========================================
```

```
hat do you want to do now?
 - Close the application
 - Check account balance
- Make a deposit
 - Withdraw money
- Transfer money from one account to another
nter your choice:
```

۷. اکنون ۰ را وارد کرده و Enter را بزنید.

۸. از پنجره ی DOS خارج گشته و به محیط برنامه نویسی بازگردید.

۹. محیط برنامه نویسی را ببندید.

متدها و دستورات شرط ها

مقداری را از متد بازگرداندن

مستحضر هستید هنگامی که متد void تعریف می کنید، مقداری بازگردانده نمی شود.

مثال

```
using System;
public class Exercise
{
  private void Show()
  {
    Console.WriteLine("C# Programming is fun!!!");
  }
  public static int Main()
  {
    Exercise exo = new Exercise();
    exo.Show();
    return 0;
  }
}
```

در حقیقت، متد void عمل بازگرداندن را انجام می دهد ولی مقداری برنمی گرداند، با این کار به compiler خبر می دهد که زمان خروج از متد فرا رسیده است (زمانی که متد void فراخوان می شود، هیچ مقدار یا نتیجه ای بازگردانده نمی شود و هیچ متغیری نمی توان تخصیص داد).

برای این منظور، در بخش مناسب متد void، کافی است کلید واژه ی return را تایپ کنید. به مثال زیر توجه کنید.

```
using System;
public class Exercise
```

```
{
  private void Show()
  {
    Console.WriteLine("C# Programming is fun!!!");
    return;
  }
  public static int Main()
  {
    Exercise exo = new Exercise();
    exo.Show();
    return 0;
  }
}
```

در این مثال، عملیات بازگشت هیچ کار خاصی انجام نمی دهد. در بخش بعدی فایده ی استفاده ی آن در دستورات شرطی را توضیح خواهیم داد.

متدها و شرط ها

۱. Microsoft Visual Studio را راه اندازی کنید.

۲. برای ایجاد برنامه ی جدید، در فهرست گزینه ی اصلی، روی File -> New Project... کلیک کنید.

۳. در فهرست میانی، روی Empty Project کلیک کنید.

۴. اسم پروژه را NationalBank5 انتخاب کرده، سپس Enter را بزنید.

۵. به منظور ایجاد فایل جدید، فهرست اصلی را بازکرده و گزینه ی Project -> Add New Item... را انتخاب کنید.

۶. Code File را از لیست میانی انتخاب کنید.

۷. اسم فایل را به Customer تغییر دهید.

۸. اکنون Add را کلیک کنید.

۹. فایل Customer.cs را به صورت زیر اصلاح کنید.

```
public enum AccountType { Checking, Saving, Other }
public class Customer
{
  public string AccountNumber;
  public AccountType Type;
  public string FullName;
  public double Balance;
  public short PIN;
  public Customer(string acnt = "000-000000-000",
          AccountType category = AccountType.Other,
          string name = "John Doe")
```

```
  {
    AccountNumber = acnt;
    Type = category;
    FullName = name;
    PIN = 0;
    Balance = 0.00D;
  }
}
```

۱۰. برای ایجاد فایل جدید، در پنجره ی Solution Explorer، راست کلیک کرده سپس : NationalBank5 -> Add -> New Item... .

۱۱. پس از انتخاب Code File از لیست میانی، اسم مورد نظر را به Management تغییر داده و کلید Enter را بزنید.

۱۲. فایل را به ترتیب زیر اصلاح کنید.

```
using System;
public class Management
{
  private Customer CreateNewAccount()
  {
    byte typeOfAccount = 0;
    Customer client = new Customer();
    Console.WriteLine("==========================================");
    Console.WriteLine("==-= National Bank =-=====================");
    Console.WriteLine("------------------------------------------");
    Console.Write("Enter a number for the new account(000-000000-000): ");
    client.AccountNumber = Console.ReadLine();
    Console.WriteLine("What type of account the customer wants to open");
    Console.WriteLine("1 - Checking Account");
    Console.WriteLine("2 - Savings Account");
    Console.Write("Enter account type: ");
    typeOfAccount = byte.Parse(Console.ReadLine());
    if (typeOfAccount == 1)
      client.Type = AccountType.Checking;
    else if (typeOfAccount == 2)
      client.Type = AccountType.Saving;
    else
      client.Type = AccountType.Other;
    Console.Write("Enter customer name: ");
    client.FullName = Console.ReadLine();
    Console.Write("Ask the customer to enter a PIN: ");
    client.PIN = short.Parse(Console.ReadLine());
    return client;
  }
  public double GetMoney()
  {
    double amount = 0;
    Console.Write("Amount: ");
    amount = double.Parse(Console.ReadLine());
    return amount;
```

```
}
private void ShowAccountInformation(Customer cust)
{
  Console.WriteLine("=============================================");
  Console.WriteLine("==-= National Bank =-=====================");
  Console.WriteLine("Customer Account Information");
  Console.WriteLine("---------------------------------------");
  Console.WriteLine("Account #:  {0}", cust.AccountNumber);
  Console.WriteLine("Account Type:{0}", cust.Type);
  Console.WriteLine("Full Name:  {0}", cust.FullName);
  Console.WriteLine("PIN #:    {0}", cust.PIN);
  Console.WriteLine("Balance:   {0:F}", cust.Balance);
  Console.WriteLine("=============================================");
  return;
}
public static int Main()
{
  double amount = 0;
  byte nextAction = 0;
  Customer accountHolder = null;
  Management registration = new Management();
  Console.Title = "National Bank";
  accountHolder = registration.CreateNewAccount();
  Console.WriteLine("Enter the customer's initial deposit");
  accountHolder.Balance = registration.GetMoney();
  Console.Clear();
  registration.ShowAccountInformation(accountHolder);
  do
  {
    Console.WriteLine("What do you want to do now?");
    Console.WriteLine("1 - Check account balance");
    Console.WriteLine("2 - Make a deposit");
    Console.WriteLine("3 - Withdraw money");
    Console.WriteLine("4 - Transfer money from one account to another");
    Console.Write("Enter your choice: ");
    nextAction = byte.Parse(Console.ReadLine());
    Console.Clear();
    switch (nextAction)
    {
      case 1:
        Console.Clear();
        registration.ShowAccountInformation(accountHolder);
        Console.Write("Press Enter for next operation");
        Console.ReadKey();
        break;
      case 2:
        Console.Write("Enter Deposit ");
        amount = registration.GetMoney();
        accountHolder.Balance = accountHolder.Balance + amount;
        Console.Clear();
```

```
            registration.ShowAccountInformation(accountHolder);
            break;
          case 3:
            Console.Write("Enter Withdrawal ");
            amount = registration.GetMoney();
            if (amount > accountHolder.Balance)
            {
              Console.WriteLine("You are not allowed to withdraw more money than your account has.");
              Console.ReadKey();
            }
            else
              accountHolder.Balance = accountHolder.Balance - amount;
            Console.Clear();
            registration.ShowAccountInformation(accountHolder);
            break;
          case 4:
            Console.WriteLine( "Operation not available: You have only one account with us");
            break;
        }
        if ((nextAction < 1) || (nextAction > 4))
          Console.WriteLine("Invalid Action: Please enter a value between 1 and 4");
      } while ((nextAction >= 1) && (nextAction <= 4));
      Console.ReadKey();
      return 0;
    }
}
```

بازگشت شرطی

بعضی از توابع با توجه به پردازشی که روی عملیات شرطی انجام می دهند مقداری را برمی گردانند. به عبارت دیگر، می توان دستور شرطی، مانند if، داخل تابع نوشت و مقداری را از شرط مزبور بازگرداند. توجه خود را به مثال زیر جلب کنید.

```
using System;
public class Program
{
  enum HouseType { Unknown, SingleFamily, Townhouse, Condominium };
  public static int Main()
  {
    var type = GetHouseType();
    switch (type)
    {
      case HouseType.SingleFamily:
        Console.WriteLine("\nType of Home: Single Family");
        break;
      case HouseType.Townhouse:
        Console.WriteLine("\nType of Home: Townhouse");
        break;
```

```
        case HouseType.Condominium:
          Console.WriteLine("\nType of Home: Condominium");
          break;
        case HouseType.Unknown:
          Console.WriteLine("\nType of Home. Unknown");
          break;
      }
      return 0;
    }
    private static HouseType GetHouseType()
    {
      var type = 0;
      Console.WriteLine("What Type of House Would you Like to Purchase?");
      Console.WriteLine("1 - Single Family");
      Console.WriteLine("2 - Townhouse");
      Console.WriteLine("3 - Condominium");
      Console.Write("Your Choice? ");
      type = int.Parse(Console.ReadLine());
      if (type == 1)
        return HouseType.SingleFamily;
      else if (type == 2)
        return HouseType.Townhouse;
      else if (type == 3)
        return HouseType.Condominium;
    }
}
```

متد GetHouseType() مشخص می کند چه زمانی یکی از سه مقدار نام برده برگردانده شده است. در حقیقت این متد می تواند مقداری به جز سه مقدار مذکور دریافت کند، اما در آن صورت (چنانچه کاربر چنین مقداری را وارد کند)، متد جاری نمی داند چه کار باید بکند. به همین دلیل، برنامه ترجمه (compile) نمی شود و در محیط Microsoft Visual C# با error زیر مواجه می شوید.

```
'Program.GetHouseType()': not all code paths return a value
```

برای حل این مشکل، باید دستوری ارائه دهید که مقداری غیر از مقدارهای درنظر گرفته را شامل شود. می توان این کار را با نوشتن return نهایی که مقدار خود را دارد انجام داد.

مثال

```
using System;
public class Program
{
  enum HouseType { Unknown, SingleFamily, Townhouse, Condominium };
  public static int Main()
  {
    var type = GetHouseType();
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    switch (type)
    {
      case HouseType.SingleFamily:
        Console.WriteLine("\nType of Home: Single Family");
        break;
      case HouseType.Townhouse:
        Console.WriteLine("\nType of Home: Townhouse");
        break;
      case HouseType.Condominium:
        Console.WriteLine("\nType of Home: Condominium");
        break;
      case HouseType.Unknown:
        Console.WriteLine("\nType of Home. Unknown");
        break;
    }
    return 0;
  }
  private static HouseType GetHouseType()
  {
    var type = 0;
    Console.WriteLine("What Type of House Would you Like to Purchase?");
    Console.WriteLine("1 - Single Family");
    Console.WriteLine("2 - Townhouse");
    Console.WriteLine("3 - Condominium");
    Console.Write("Your Choice? ");
    type = int.Parse(Console.ReadLine());
    if (type == 1)
      return HouseType.SingleFamily;
    else if (type == 2)
      return HouseType.Townhouse;
    else if (type == 3)
      return HouseType.Condominium;
    else
      return HouseType.Unknown;
  }
}
```

بازگرداندن مقداری به صورت شرطی

۱. فایل Management.cs را بدین ترتیب تغییر دهید.

```
using System;
public class Management
{
  private AccountType SpecifyAccountType()
  {
    byte type = 0;
    Console.WriteLine("What type of account the customer wants to open");
    Console.WriteLine("1 - Checking Account");
```

```
    Console.WriteLine("2 - Savings Account");
    Console.Write("Enter account type: ");
    type = byte.Parse(Console.ReadLine());
    if (type == 1)
      return AccountType.Checking;
    else if (type == 2)
      return AccountType.Saving;
    else
      return AccountType.Other;
  }
  private Customer CreateNewAccount()
  {
    Customer client = new Customer();
    . . . No Change
    return client;
  }
  public double Deposit()
  {
    double amount = 0;
    Console.Write("Amount to deposit: ");
    amount = double.Parse(Console.ReadLine());
    return amount;
  }
  public double Withdraw(Customer cust)
  {
    double amount = 0;
    Console.Write("Amount to withdraw: ");
    amount = double.Parse(Console.ReadLine());
    if (amount > cust.Balance)
    {
      Console.WriteLine(
      "You are not allowed to withdraw more money than your account has.");
      Console.ReadKey();
      return 0.00D;
    }
    else
      return amount;
  }
  private void ShowAccountInformation(Customer cust)
  {
    . . . No Change
  }
  public static int Main()
  {
    double amount = 0;
    byte nextAction = 0;
    Customer accountHolder = null;
    Management registration = new Management();
    Console.Title = "National Bank";
    accountHolder = registration.CreateNewAccount();
```

```
  Console.WriteLine("Enter the customer's initial deposit");
  accountHolder.Balance = registration.Deposit();
  Console.Clear();
  registration.ShowAccountInformation(accountHolder);
  do
  {
    Console.WriteLine("What do you want to do now?");
    Console.WriteLine("0 - Close the application");
    Console.WriteLine("1 - Check account balance");
    Console.WriteLine("2 - Make a deposit");
    Console.WriteLine("3 - Withdraw money");
    Console.WriteLine("4 - Transfer money from one account to another");
    Console.Write("Enter your choice: ");
    nextAction = byte.Parse(Console.ReadLine());
    Console.Clear();
    switch (nextAction)
    {
      case 1:
        Console.Clear();
        registration.ShowAccountInformation(accountHolder);
        Console.Write("Press Enter for next operation");
        Console.ReadKey();
        break;
      case 2:
        amount = registration.Deposit();
        accountHolder.Balance += amount;
        Console.Clear();
        registration.ShowAccountInformation(accountHolder);
        break;
      case 3:
        amount = registration.Withdraw(accountHolder);
        accountHolder.Balance -= amount;
        Console.Clear();
        registration.ShowAccountInformation(accountHolder);
        break;
      case 4:
        Console.WriteLine(
       "Operation not available: You have only one account with us");
        break;
      default:
        return 0;
    }
    if ((nextAction < 1) || (nextAction > 4))
      Console.WriteLine(
      "Invalid Action: Please enter a value between 1 and 4");
  } while ((nextAction >= 1) && (nextAction <= 4));
  Console.ReadKey();
  return 0;
}
 }
```

۲. کلید F5 را زده تا برنامه اجرا شود.

۳. اطلاعات لازم را به این ترتیب وارد کنید.

Account #:	257-484902-444
Account Type:	1
Customer Name:	Josh Nay
PIN:	5008
Initial Deposit	250

```
==========================================
=-= National Bank =-=====================
------------------------------------------
ew Account Number (000-000000-000): 257-484902-444
hat type of account the customer wants to open
 - Checking Account
 - Savings Account
nter account type: 1
nter customer name: Josh Nay
sk the customer to enter a PIN: 5008
nter the customer's initial deposit
mount to deposit: 250
```

۴. کلید Enter را فشار دهید.

```
==========================================
==-= National Bank =-=====================
Customer Account Information
------------------------------------------
Account #:  257-484902-444
Account Type: Checking
Full Name:  Josh Nay
PIN #:  5008
Balance:  250.00
==========================================
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

What do you want to do now?
0 - Close the application
1 - Check account balance
2 - Make a deposit
3 - Withdraw money
4 - Transfer money from one account to another
Enter your choice:

۵. حال به منظور برداشت مبلغ مورد نظر از حساب 3 را وارد کرده، سپس Enter را بزنید.

۶. 164.37 را به عنوان مبلغ واریزی تایپ کنید و کلید Enter را بزنید.

==========

==-= National Bank =-==========

Customer Account Information

Account #: 257-484902-444

Account Type: Checking

Full Name: Josh Nay

PIN #: 5008

Balance: 85.63

==========

What do you want to do now?

0 - Close the application

1 - Check account balance

2 - Make a deposit

3 - Withdraw money

4 - Transfer money from one account to another

Enter your choice:

۷. اکنون عدد 2 را وارد کنید و Enter را بزنید.

۸. مبلغ سپرده را 116.18 وارد کنید، سپس Enter را بزنید.

==========

==-= National Bank =-==========

Customer Account Information

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

Account #: 257-484902-444

Account Type: Checking

Full Name: Josh Nay

PIN #: 5008

Balance: 201.81

==========

What do you want to do now?

0 - Close the application

1 - Check account balance

2 - Make a deposit

3 - Withdraw money

4 - Transfer money from one account to another

Enter your choice:

۹. در این مرحله، عدد 3 را وارد کنید و کلید Enter را بزنید.

۱۰. رقم 300 را به عنوان مبلغ برداشتی وارد کنید، سپس Enter را بزنید.

Amount to withdraw: 300

You are not allowed to withdraw more money than your account has.

۱۱. کلید Enter را بزنید.

۱۲. عدد ۰ را وارد کرده و دوباره Enter را بزنید.

While (true)

تاکنون در موقعیت های مختلفی که شرط while را بکار بردیم، وسیله ای برای بررسی شرط مورد نظر نیز گنجاندیم. به جای این کار همچنین می توان ثابت صحیح بولی را داخل پرانتزهای مقدار true قرار داد.

مثال

```
using System;
public class Exercise
{
  public static int Main()
  {
    while (true)
```

```
        Console.WriteLine("C# Programming is fun!!!");
      return 0;
    }
  }
```

این نوع دستور معتبر بوده و کار می کند ولی هیچ راهی برای متوقف کردن آن وجود ندارد، زیرا دستور نام برده به کامپایلر می گوید " تا زمانی که این درست است ("as long as this is true"،...") ". سوال مطرح این است که منظور از "this" چیست ؟ در نتیجه، برنامه بدون هیچ گونه توقف و برای همیشه ادامه پیدا می کند. بنابراین، در صورت ایجاد شرط while(true)، در بدنه ی دستور مزبور، لازم است وسیله ای برای متوقف کردن کامپایلر فراهم کرد (به عبارت دیگر، راهی که از آن طریق شرط به false ارزیابی شود). برای نیل به این هدف، باید شرط if را به کار برد.

مثال

```
using System;
public class Exercise
{
  public static int Main()
  {
    int i = 0;
    while (true)
    {
      if (i > 8)
        break;
      Console.WriteLine("C# Programming is fun!!!");
      i++;
    }
    return 0;
  }
}
```

این بار برنامه از compiler درخواست می کند جمله ای را در صفحه ی کنسول نمایش دهد که از ۰ تا ۸ می شمارد ولی بمحض رسیدن به ۸ متوقف شود. برنامه ی فوق این نتیجه را به دست می دهد.

```
C# Programming is fun!!!
C# Programming is fun!!!
C# Programming is fun!!!
C# Programming is fun!!!
C# Programming is fun!!!
C# Programming is fun!!!
C# Programming is fun!!!
```

```
C# Programming is fun!!!
C# Programming is fun!!!
Press any key to continue…
```

در برنامه ی بالا، از عملگر break استفاده کردیم. البته می توان از هر مکانیزم دیگری برای متوقف کردن حلقه کمک گرفت مادام اینکه شرط مورد نظر false شود. مثال زیر شیوه ی دیگری از متوقف کردن حلقه را به نمایش می گذارد.

```
using System;
public class Exercise
{
  public static int Main()
  {
    int i = 0;
    while (true)
    {
      if (i < 8)
        Console.WriteLine("C# Programming is fun!!!");
      else
        return 0;
      i++;
    }
  }
}
```

نتیجه ی آن با مثال پیشین یکسان می باشد.

به جای استفاده از while(true)، می توان ابتدا متغیر بولی را تعریف و مقداردهی اولیه کرد، یا می توان متغیر بولی به کاربرد که مقدار آن از پیش شناس و مشخص است. مقدار مربوط می تواند از یک متد بیاید یا از هر وسیله ی دیگری.

استفاده از while (true)

۱. برای راه اندازی برنامه (کاربردی) جدید، به فهرست گزینه ی اصلی مراجعه کرده و File -> New Project… را کلیک کنید.

۲. گزینه ی Empty Project را از لیست میانی انتخاب کنید.

۳. اسم پروژه را به DepartmentStore8 تغییر دهید.

۴. به منظور ایجاد کلاس جدید، در فهرست اصلی، روی Project -> Add Class… کلیک کنید.

۵. اسم کلاس را StoreItem انتخاب کرده، سپس Add را کلیک کنید.

۶. فایل را به ترتیب زیر اصلاح کنید.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace DepartmentStore8
{
  public class StoreItem
  {
    public int itemNumber;
    public string itemName;
    public string itemSize;
    public decimal unitPrice;
    public StoreItem()
    {
      itemNumber = 0;
      itemName = "";
      itemSize = "";
      unitPrice = 0.00M;
    }
    public StoreItem(int code, string name,
            string size, decimal price)
    {
      itemNumber = code;
      itemName = name;
      itemSize = size;
      unitPrice = price;
    }
  }
}
```

۷. در پنجره ی Solution Explorer، راست کلیک کرده سپس : DepartmentStore8 -> Add -> New Item...

۸. گزینه ی Code File را از لیست میانی انتخاب کنید.

۹. اسم آن را به Inventory تغییر داده و Enter را بزنید.

۱۰. فایل را به ترتیب زیر اصلاح کنید.

```
using System;
using DepartmentStore8;
public class Inventory
{
  static int Main()
  {
    StoreItem si = null;
    bool itemIsValid = true;
    Console.Title = "Department Store Inventory";
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    Console.WriteLine("Department Store Inventory");
    while (itemIsValid)
    {
      si = new StoreItem();
      Console.WriteLine("------------------------");
      Console.Write("Enter Item #:  ");
      si.itemNumber = int.Parse(Console.ReadLine());
      if (si.itemNumber < 0)
        itemIsValid = false;
      else
      {
        Console.Write("Enter Item Name: ");
        si.itemName = Console.ReadLine();
        Console.Write("Enter Item Size: ");
        si.itemSize = Console.ReadLine();
        Console.Write("Enter Unit Price: ");
        si.unitPrice = decimal.Parse(Console.ReadLine());
        Console.WriteLine("=-= Store Item =-=");
        Console.WriteLine("Item #:  {0}", si.itemNumber);
        Console.WriteLine("Item Name: {0}", si.itemName);
        Console.WriteLine("Size:  {0}", si.itemSize);
        Console.WriteLine("Unit Price: {0}", si.unitPrice);
      }
    }
    return 0;
  }
}
```

۱۱. برنامه را تست کنید.

۱۲. اطلاعات مورد نیاز را به صورت زیر وارد کنید.

Item Number:	**279475**
Item Name:	Wool Jersey Ruffle Dress
Size:	8
Unit Price:	150
Item Number:	729070
Item Name:	Mid-Weight Flat-Front Wool Trouser Pants

Size:	36W - 30L
Unit Price:	69.95
Item Number:	297004
Item Name:	Ponte Pencil Skirt with Pleated Detail
Size:	16
Unit Price:	80
Item Number:	-1

Department Store Inventory

Enter Item #: 279475

Enter Item Name: Wool Jersey Ruffle Dress

Enter Item Size: 8

Enter Unit Price: 150

=-= Store Item =-=

Item #: 279475

Item Name: Wool Jersey Ruffle Dress

Size: 8

Unit Price: 150

Enter Item #: 729070

Enter Item Name: Mid-Weight Flat-Front Wool Trouser Pants

Enter Item Size: 36W - 30L

Enter Unit Price: 69.95

=-= Store Item =-=

Item #: 729070

Item Name: Mid-Weight Flat-Front Wool Trouser Pants

Size: 36W - 30L

Unit Price: 69.95

Enter Item #: 297004

Enter Item Name: Ponte Pencil Skirt With Pleated Detail

Enter Item Size: 16

```
Enter Unit Price: 80
=-= Store Item =-=
Item #:  297004
Item Name: Ponte Pencil Skirt With Pleated Detail
Size:  16
Unit Price: 80
-----------------------
Enter Item #:  -1
```

۱۳. پنجره ی DOS را بسته و به محیط برنامه نویسی بازگردید.

بازگشت

تصور کنید می خواهید اعداد فرد مثبت را از بیشنه مشخصی تا کمینه ی معینی بشمارید. به عنوان مثال، برای شمردن اعداد فرد از ۱ تا ۹ به این ترتیب عمل می کنیم.

1، 3، 5، 7،9

توجه داشته باشید که برای انجام این عملیات، بالاترین مقدار را در نظر می گیریم، سپس ۲ را از آن کم می کنیم تا مقدار قبلی را به دست آوریم. در برنامه نویسی برای حل این مشکل، ابتدا تابع را می نویسیم سپس کاری می کنیم که تابع خود را فرابخواند. این پایه ی بازگشت در برنامه نویسی است.

معرفی بازگشت

۱. برای راه اندازی برنامه ی کاربردی جدید، در فهرست گزینه ی اصلی، File -> New Project... را کلیک کنید.

۲. Empty Project را از لیست میانی انتخاب کنید.

۳. اسم پروژه را به Recursions تغییر دهید، کلید Enter را بزنید.

۴. در پنجره ی Solution Explorer، راست کلیک کرده سپس : Recursions -> Add -> New Item

۵. در لیست میانی روی Code File کلیک کنید.

۶. اسم فایل را Calculator انتخاب کنید و کلید Enter را فشار دهید.

ایجاد متد بازگشتی

یکی از فرمول های موجود برای ایجاد متد بازگشتی به صورت زیر می باشد.

```
ReturnValue Function(Arguments, if any)
{
  Optional Action . . .
  Function();
  Optionan Action . . .
}
```

متد بازگشتی با یک مقدار بازگشتی شروع می شود. در صورت برنگرداندن یک مقدار، می توان آن را با کلید واژه ی void تعریف کرد. پس از اسمش، متد قادر است یک یا چند آرگومان بگیرد. اغلب مواقع، متد بازگشتی حداقل یک آرگومان می گیرد، سپس آن را اصلاح می کند. در بدنه ی متد، می توان هر کاری که لازم است انجام داد. دو قاعده ی اصلی وجود دارد که حین پیاده سازی متد بازگشتی باید رعایت کرد.

متد باید خود را در بدنه ی خود فراخوانی کند.

پیش یا پس از فراخوانی خود، متد باید شرطی را بررسی کند که به متد اجازه ی توقف بدهد. در غیر این صورت، متد تا ابد ادامه پیدا می کند.

برای مثال بالا (منظور مثال بازگشت می باشد)، می توان ابتدا متدی ایجاد کرد که عدد صحیح (integer) به عنوان آرگومان می گیرد. به منظور درک بهتر این مطلب، فقط اعداد مثبت را در نظر می گیریم. در بدنه ی متد، مقدار جاری آرگومان را نمایش می دهیم، سپس ۲ واحد از آن کم می کنیم و در مرحله ی پایانی خود متد را فراخوانی می کنیم.

```
using System;
public class Exercise
{
  static void OddNumbers(int a)
  {
    if (a >= 1)
    {
      Console.Write("{0}, ", a);
      a -= 2;
      OddNumbers(a);
    }
  }
  public static int Main()
  {
    const int number = 9;
    Console.WriteLine("Odd Numbers");
    OddNumbers(number);
    Console.WriteLine();
    return 0;
  }
}
```

همان طور که مشاهده می کنید متد خود را در بدنه اش فرا می خواند. نتیجه ی زیر به دست می آید.

```
Odd Numbers
 9، 7، 5، 3، 1،
 Press any key to continue…
```

ایجاد متد بازگشتی

۱. برای ایجاد تابع بازگشتی، ابتدا فایل را به صورت زیر اصلاح کنید.

```
using System;
 public class Calculator
 {
   private long Factorial(long number)
   {
     if (number <= 1)
       return 1;
     return number * Factorial(number - 1);
   }
   public static int Main()
   {
     long factor = 0;
     Calculator exo = new Calculator();
     Console.Write("To calculate a factorial, enter a (small positive) number: ");
     factor = long.Parse(Console.ReadLine());
     Console.WriteLine("The factorial of {0} = {1}",factor, exo.Factorial(factor));
     System.Console.ReadKey();
     return 0;
   }
 }
```

۲. برنامه را تست کنید.

۳. عدد ۸ را وارد کرده و Enter را بزنید.

```
To calculate a factorial، enter a (small positive) number: 8
 The factorial of 8 = 40320
 Press any key to continue…
```

۴. با زدن کلید Enter از پنجره ی DOS خارج شوید.

استفاده از متدهای بازگشتی

متد های بازگشتی مکانیزم های ارزشمندی برای ساختن سری ها و لیست ها عرضه می کنند که در حقیقت مقادیر افزایشی یا کاهشی هستند که از یک الگو پیروی می کنند. تصور کنید به جای تنها نمایش دادن اعداد فرد (همان طور که بالا نشان دادیم)، می خواهیم آن ها را به صورت تصاعدی / افزایشی اضافه کنیم. در صورت داشتن ۱، ۱ تولید می کند. اگر ۵ دارید و ۱ را به ۳ اضافه می کنید و بعد نتیجه ی آن را به ۵ (اضافه کنید) و غیره.... این فرایند را می توان به ترتیب زیر نمایش داد.

1 = 1
1 + 3 = 4
1 + 3 + 5 = 9
1 + 3 + 5 + 7 = 16
1 + 3 + 5 + 7 + 9 = 25

چنانچه عدد مساوی ۱ یا کوچک تر از آن بود، متد ۱ باز می گرداند. در غیر این صورت، ۲ را . برای اجرای این عملیات، ابتدا ۱ را در نظر می گیریم. به ۱ اضافه می کنیم، سپس ۲ را به نتیجه ی حاصله (۱+۲) اضافه می کنیم. این پروسه را ادامه دهید تا به مقدار آرگومان برسید. متد به ترتیب زیر ایجاد یا پیاده سازی می شود.

```
using System;
 public class Exercise
{
   static int AdditionalOdd(int a)
   {
     if (a <= 1)
       return 1;
     return a + AdditionalOdd(a - 2);
   }
   static void OddNumbers(int a)
   {
     if (a >= 1)
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    {
      Console.Write("{0}, ", a);
      a -= 2;
      OddNumbers(a);
    }
  }
  public static int Main()
  {
    const int Number = 9;
    Console.WriteLine("Odd Numbers");
    OddNumbers(Number);
    Console.WriteLine();
    Console.WriteLine("Sum of Odds: {0}\n", AdditionalOdd(Number));
    return 0;
  }
}
```

نتیجه ی زیر حاصل می گردد.

```
Odd Numbers
9, 7, 5, 3, 1,
Sum of Odds: 25
Press any key to continue...
```

به کاربردن متدهای بازگشتی

۱. فایل را به ترتیب زیر اصلاح کنید.

```
using System;
public class Calculator
{
  private long Factorial(long number)
  {
    if (number <= 1)
      return 1;
    return number * Factorial(number - 1);
  }
  private long Permutation(long n, long r)
  {
    if (r == 0)
      return 0;
    if (n == 0)
      return 0;
    if ((r >= 0) && (r <= n))
      return Factorial(n) / Factorial(n - r);
    else
      return 0;
  }
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

```
    private long Combinatorial(long a, long b)
    {
      if (a <= 1)
        return 1;
      return Factorial(a) / ((Factorial(b) * Factorial(a - b)));
    }
    public static int Main()
    {
      long factor = 0;
      long second = 0;
      Calculator exo = new Calculator();
      Console.Write("To calculate a factorial, enter a (small positive) number: ");
      factor = long.Parse(Console.ReadLine());
      Console.Write("To calculate a permutation and the combination, enter a second (small positive) number: ");
      second = long.Parse(Console.ReadLine());
      Console.WriteLine("Factorial:  F({0}) = {1}",factor, exo.Factorial(factor));
      Console.WriteLine("Permutation: P({0}, {1}) = {2}", factor, second, exo.Permutation(factor, second));
      Console.WriteLine("Combination: C({0}, {1}) = {2}",factor, second, exo.Combinatorial(factor, second));
      System.Console.ReadKey();
      return 0;
    }
}
```

۱. برنامه را تست کنید.

۲. به عنوان اولین عدد درخواستی، 20 را وارد کنید و بعد Enter را بزنید.

۳. به عنوان عدد دوم، 5 را وارد کرده و کلید Enter را فشار دهید.

```
To calculate a factorial, enter a (small positive) number: 20
To calculate a permutation and the combination, enter a second (small positive)
 number: 5
Factorial:  F(20) = 2432902008176640000
Permutation: P(20, 5) = 1860480
Combination: C(20, 5) = 15504
```


آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

در پایان ضمن تشکر از انتخاب شما،امیدواریم مطالب این کتاب برای شما مفید بوده باشد.

علاوه بر این می توانید پیشنهادات و انتقادات خود را از طریق رایانامه Book.tahlildadeh@gmail.com با ما در میان بگذارید.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 طبقه دوم - واحد7
88146323 - 88446780 - 88146330