

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِیْمِ

برنامه‌سازی ۲

رشته کامپیوتر

گروه تحصیلی کامپیوتر

زمینه خدمات

شاخه آموزش فنی و حرفه‌ای

عنوان و نام پدیدآور	:	برنامه‌سازی ۲ : رشته کامپیوتر گروه تحصیلی کامپیوتر زمینه خدمات شاخه آموزش فنی و حرفه‌ای.
مشخصات نشر	:	تهران : شرکت چاپ و نشر کتاب‌های درسی ایران، ۱۳۹۵
مشخصات ظاهری	:	۲۱۹ ص. : مصور.
شابک	:	۹۷۸-۹۶۴-۰۵-۲۴۱۳-۸
وضعیت فهرست‌نویسی	:	فیبای مختصر
شناسه افزوده	:	کربلایی، مجید
شناسه افزوده	:	سازمان پژوهش و برنامه‌ریزی آموزشی. دفتر تألیف کتاب‌های درسی فنی و حرفه‌ای و کار دانش
شماره کتاب‌شناسی ملی	:	۳۵۹۳۹۲۱



از شماست که مردان و زنان بزرگ تربیت می شود. شماها در تحصیل کوشش کنید که برای فضایل اخلاقی، فضایل اعمالی مجتهد شوید. شما برای آتیه مملکت ما جوانان نیرومند تربیت کنید. دامن شما یک مدرسه ای است که در آن جوانان بزرگ تربیت بشود. شما فضایل تحصیل کنید تا کودکان شما در دامن شما به فضیلت برسند.

امام خمینی «ره»

فهرست

اول یادآوری و تکمیل مطالب

- ۱-۱ یادآوری ۲
- ۱-۲ استفاده از محیط توسعه برنامه متمرکز ۳
- ۱-۳ یادآوری دستورات زبان برنامه‌نویسی C# ۱۱
- ۱-۴ استفاده از مقدارثابت در برنامه ۲۹
- ۱-۵ حلقه‌های متداخل (تو در تو) ۳۰
- ۱-۶ کاراکتر \ ۳۴
- خودآزمایی ۳۷
- تمرینات برنامه‌نویسی ۳۹
- فعالیت ۴۳
- واژگان و اصطلاحات انگلیسی ۴۴

دوم آرایه

- ۲-۱ تعریف آرایه ۴۶
- ۲-۲ نحوه ایجاد یا تعریف آرایه در زبان C# ۴۷
- ۲-۳ دسترسی، مقداردهی و نمایش عناصر آرایه ۴۸
- ۲-۴ حلقه Foreach ۵۳
- ۲-۵ تعیین اندازه آرایه در هنگام اجرای برنامه ۵۶
- ۲-۶ مرتب کردن داده‌های یک لیست ۵۸
- ۲-۷ عمل جستجو در لیست ۶۳
- خودآزمایی ۶۹
- تمرینات برنامه‌نویسی ۷۲
- فعالیت ۷۳
- واژگان و اصطلاحات انگلیسی ۷۴

سوم داده‌های شمارشی، کلاس و متد

- ۳-۱- نوع داده شمارشی ۷۶
- ۳-۲- استفاده از نوع داده‌های شمارشی آماده در کتابخانه Net ۸۴
- ۳-۳- شیء ۸۵
- خودآزمایی ۹۴
- فعالیت ۹۶
- واژگان و اصطلاحات انگلیسی ۹۷

چهارم کار با متدها و کلاس‌های آماده

- ۴-۱- کلاس Math ۹۹
- ۴-۲- کلاس String ۱۰۵
- ۴-۳- کلاس Array ۱۰۹
- خودآزمایی ۱۱۳
- تمرینات برنامه‌نویسی ۱۱۵
- فعالیت ۱۱۶
- واژگان و اصطلاحات انگلیسی ۱۱۷

پنجم ایجاد برنامه‌های ویندوزی با استفاده از ویژوال استودیو

- ۵-۱- تفاوت برنامه‌های کنسولی و برنامه‌های ویندوزی ۱۱۹
- ۵-۲- واسط گرافیکی کاربر ۱۲۲
- ۵-۳- ایجاد یک پروژه ویندوزی با کمک VS ۱۲۳
- ۵-۴- ساخت یک واسط کاربری با استفاده از قابلیت طراحی تصویری در VS ۱۳۷
- خودآزمایی ۱۵۸
- تمرینات برنامه‌نویسی ۱۶۰
- فعالیت ۱۶۱
- واژگان و اصطلاحات انگلیسی ۱۶۲

- ۱-۶- برنامه ویندوزی ساده با واکنش به رویداد کلیک ۱۶۴
- ۲-۶- کنترل جعبه متن ۱۸۶
- ۳-۶- کنترل زمان‌سنج ۱۹۴
- ۴-۶- کنترل کادر محاوره‌ای انتخاب فایل ۱۹۸
- ۵-۶- کنترل کادر محاوره‌ای رنگ و کادر محاوره‌ای فونت ۲۰۲
- ۶-۶- کنترل کادر علامت ۲۰۵
- خودآزمایی ۲۱۰
- تمرینات برنامه‌نویسی ۲۱۱
- فعالیت ۲۱۳
- واژگان و اصطلاحات انگلیسی ۲۱۴

مقدمه

زبان برنامه‌نویسی C# یکی از زبان‌های برنامه‌نویسی متداول و محبوب در بین برنامه‌نویسان می‌باشد. در کتاب برنامه‌سازی ۱ با برخی از جنبه‌های مقدماتی این زبان آشنا شدید. در این کتاب با جنبه‌های پیشرفته این زبان آشنا می‌شوید. یادگیری مفاهیم این کتاب شما را قادر خواهد ساخت تا برنامه‌های کامل‌تر و پیچیده‌تر نوشته و امکانات این زبان را به نحو مناسب‌تری به کار ببرید.

فصل اول کتاب به یادآوری و مرور مطالب کتاب برنامه‌سازی ۱ اختصاص دارد. البته در این فصل علاوه بر مرور مطالب کتاب قبلی، چند مطلب جدید نیز می‌آموزید. فصل ۲ به مفهوم مهم آرایه در زبان C# می‌پردازد. مفهوم آرایه در تمامی زبان‌های برنامه‌سازی کاربرد بسیار مهمی دارد. فصل ۳ به کلاس و متد اختصاص دارد که شروع شیء‌گرایی می‌باشد و شما را با دنیای جالب برنامه‌نویسی شیء‌گرا آشنا می‌سازد. البته مفاهیم پیشرفته‌تر شیء‌گرایی را در کتاب برنامه‌سازی ۳ فرا خواهید گرفت. در فصل ۴ یاد می‌گیرید که چگونه با متدها و کلاس‌های آماده زبان C# کار کنید. در فصل ۵ اصول ساخت یک برنامه ویندوزی با زبان C# را فرا خواهید گرفت و در نهایت در فصل ۶، با نحوه کار کنترل‌های مختلف آشنا خواهید شد.

در همه فصل‌ها سعی شده با آوردن فعالیت‌های کارگاهی، مطلب ارائه شده در قالب پروژه‌های مختلف به صورت کاملاً عملی آموزش داده شود. لذا از هنرجویان انتظار می‌رود با انجام تمرینات کار در کارگاه و خودآزمایی‌ها و فعالیت‌های آورده شده، زمینه یادگیری مطالب را فراهم نمایند.

در پایان از تمام دوستانی که در تألیف این کتاب یاری‌رسان اینجانب بوده‌اند، از جمله سرکار خانم مهناز کارکن، سرکار خانم زهرا عسگری رکن‌آبادی، جناب آقای حسین صفوی‌تاشکر و قدردانی می‌نمایم. ایده‌ها و پیشنهادات شما باعث بهبود کیفیت کتاب خواهد شد. لذا مواردی که به نظر شما می‌رسد را به دیده منت می‌نهمم و ارجح می‌گذاریم.

مجید کربلایی

یادآوری و تکمیل مطالب

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

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

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

۱-۱- یادآوری

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

مثال ۱-۱: برنامه ای بنویسید که از لیست زبان های نمایش داده شده، زبان سطح میانی را تشخیص دهد و انتخاب نماید.

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

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

برنامه ۱-۱- تشخیص زبان سطح میانی

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

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

```
CSC Test.cs
```

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

```
Test.EXE
```

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

۲-۱- استفاده از محیط توسعه برنامه متمرکز

در برنامه‌های کنسولی ابتدا با استفاده از یک ویرایشگر^۱ متن مانند Notepad، برنامه را نوشته و سپس با پسوند cs. ذخیره کرده و با مترجم C# یعنی برنامه CSC.EXE کامپایل و اجرا می‌کردیم. روش دیگری که می‌توان برای تولید فایل اجرایی برنامه استفاده کرد، استفاده از یک محیط برنامه‌نویسی متمرکز (IDE^۲) است که برای نوشتن و عیب‌یابی راحت‌تر برنامه‌ها، طراحی شده است. چندین IDE به وسیله شرکت‌ها و ارگان‌های مختلف طراحی و به بازار عرضه شده است. بعضی از آنها رایگان و به اصطلاح متن‌باز^۳ و بعضی دیگر تجاری^۴ هستند. برای انتخاب IDE باید با توجه به سیستم عامل کامپیوتر و بودجه مورد نظر اقدام کرد. مثلاً در سیستم عامل ویندوز می‌توان از محیط برنامه‌نویسی رایگان و متن‌باز SharpDevelop یا به اختصار #develop^۵ استفاده نمود (شکل ۱-۱).

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

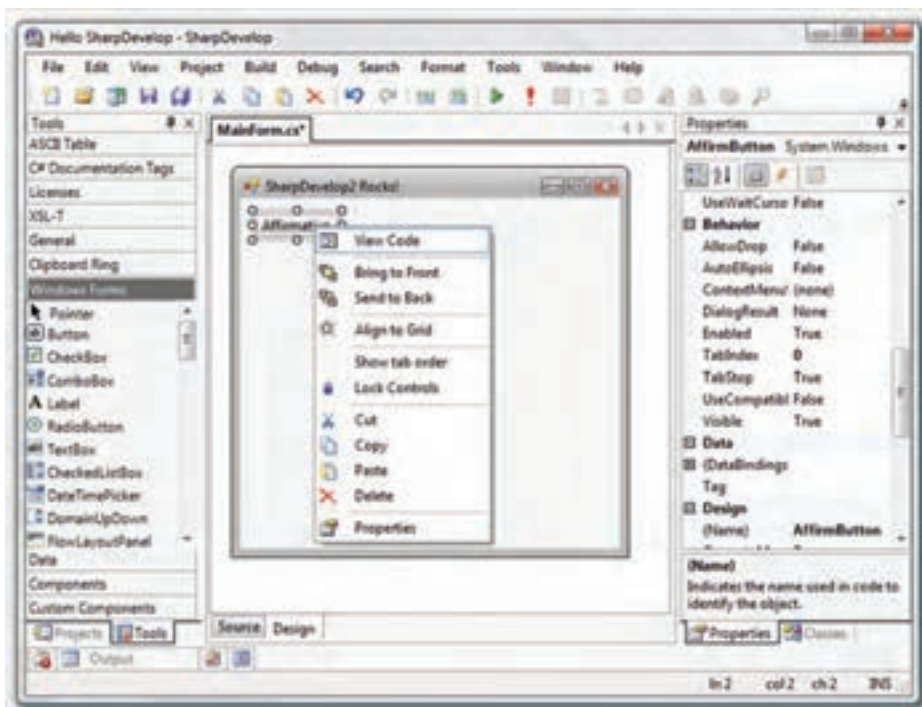
۱- Editor

۲- Integrated Development Environment

۳- Open Source

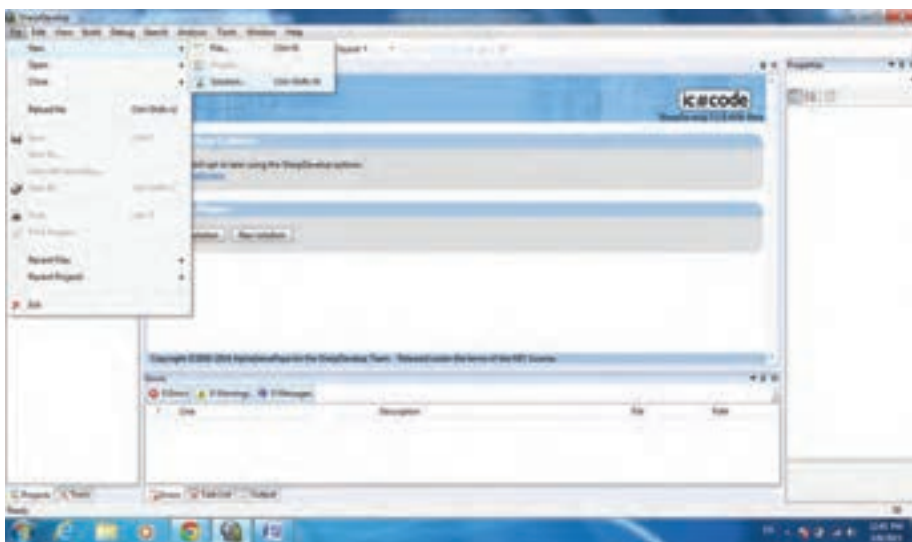
۴- Commercial

۵- # develop (short for SharpDevelop)



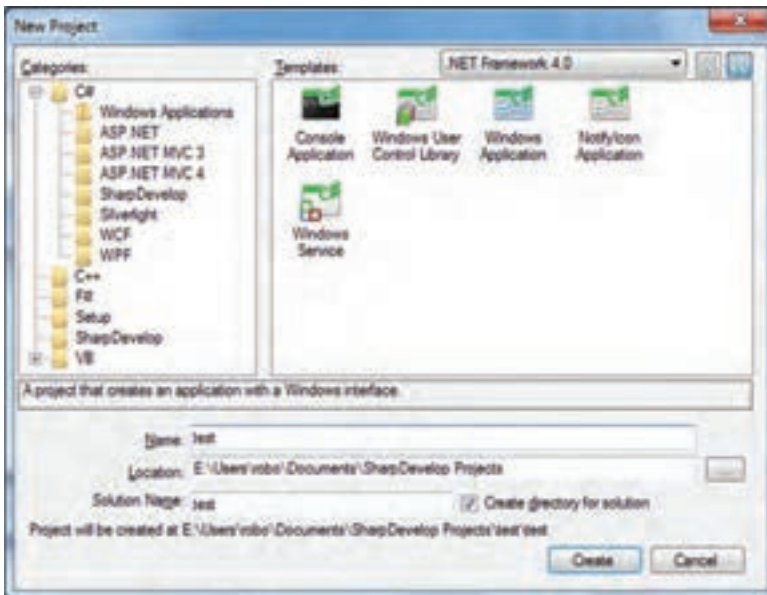
شکل ۱-۱- محیط برنامه نویسی رایگان و متن باز SharpDevelop

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



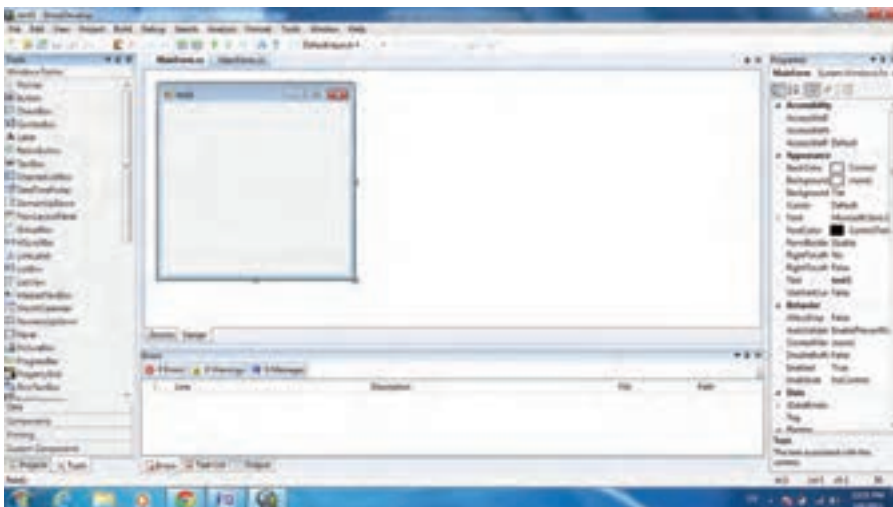
شکل ۱-۲- انتخاب گزینه solution از زیر منوی new

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



شکل ۳-۱- تعیین نام پروژه

روی گزینه create کلیک می‌کنیم. با انتخاب برگه design و برگه tools می‌توانیم به فرم اصلی و ابزارهای برنامه نویسی دسترسی داشته باشیم.



۵

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

در این IDE فرم ابتدایی MainForm نام دارد.

تاکنون از محیط برنامه نویسی ویژوال استودیو اکسپرس^۱ نسخه ۲۰۱۲ استفاده می‌کردید که مایکروسافت آن را به طور رایگان از طریق سایت خود عرضه می‌کند، تا برنامه نویسان برای تولید برنامه و ارزیابی این محصول، از آن استفاده کنند و نهایتاً نسخه اصلی ویژوال استودیو را خریداری^۲ نمایند.



شکل ۵-۱- محیط Visual Studio

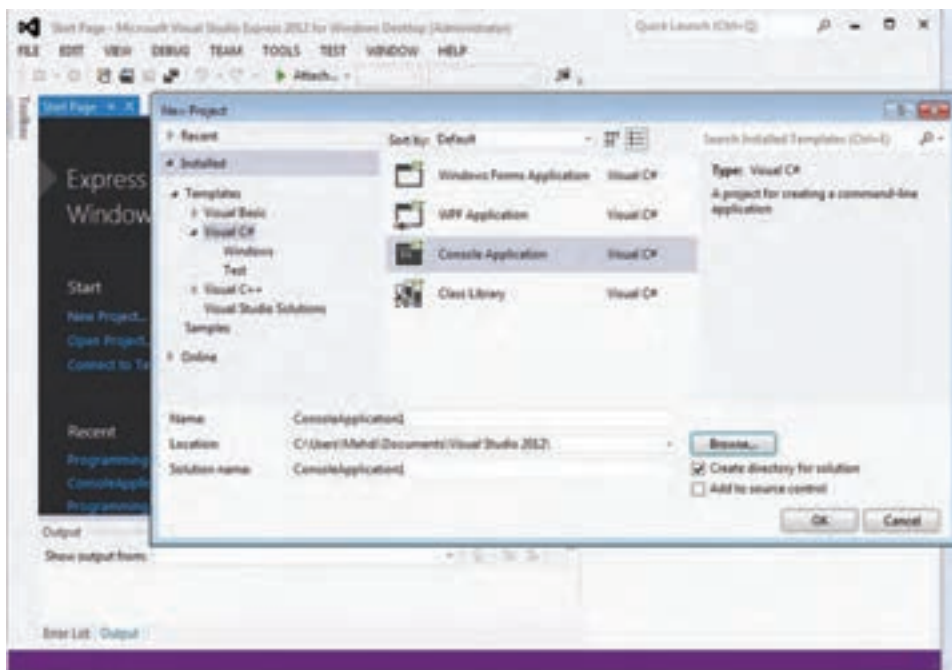
برای نوشتن برنامه‌های این کتاب نیز از ویژوال استودیو اکسپرس ۲۰۱۲ یا به اختصار VS^۳، استفاده می‌کنیم. در اینجا طریقه نوشتن برنامه و اجرای آن در VS را به اختصار مرور می‌کنیم. برای نوشتن برنامه به زبان C# مانند برنامه ۱-۱، ابتدا VS را اجرا کرده، و در پنجره Start page گزینه New Project، گزینه Visual C# و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

۱- Visual Studio Express

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

مراجعه کنید).

۳- Visual Studio



شکل ۱-۶- پنجره پروژه جدید

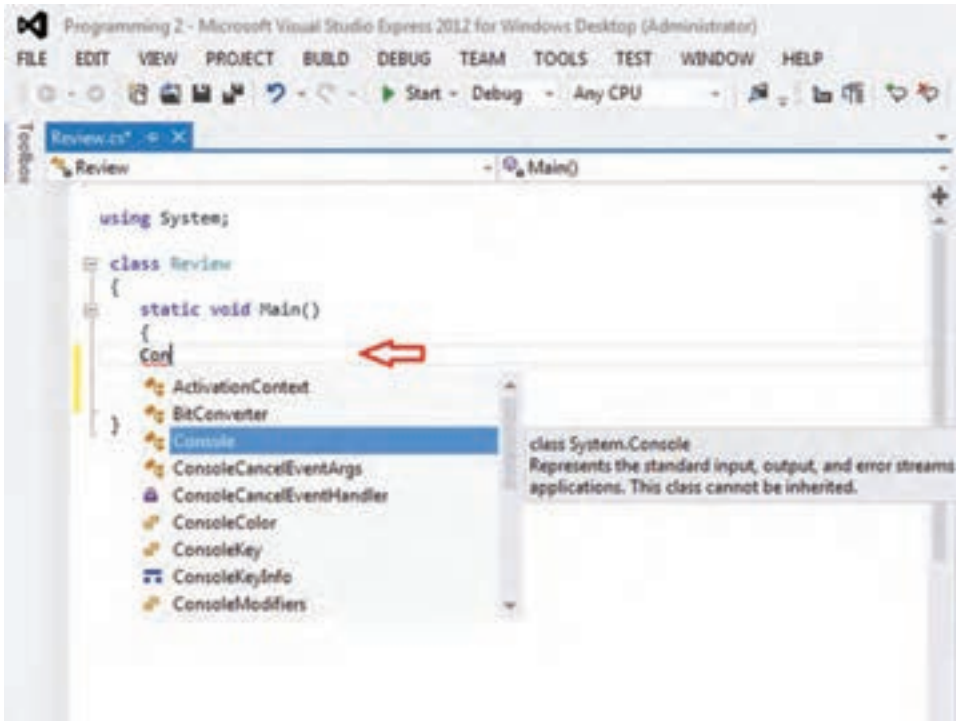
پس از مشخص کردن نام پروژه و مسیری که در آن پروژه ذخیره می‌شود، بر روی OK کلیک می‌کنیم و در پنجره ویرایشگر برنامه، دستورات را در داخل Main() می‌نویسیم.

محیط VS بسیار قدرتمند و دارای ابزارهای مختلفی برای نوشتن، اجرا و عیب‌یابی برنامه است. یکی از امکانات ارزشمند در هنگام نوشتن برنامه، IntelliSense^۱ است که در هنگام تایپ دستورات برنامه، با نوشتن چند حرف از نام دستور (نام کلاس، متد، ...)، منو یا لیستی ظاهر می‌شود که در آن نام متدها و دستورات مرتبط نشان داده شده‌اند. در این لیست می‌توانید با کلیدهای فلش بالا و پایین، حرکت کنید و سپس کلمه مورد نظر خود را برگزینید. با زدن کلید Tab یا کلید فاصله^۲، دستور مورد نظر، در برنامه به طور خودکار قرار می‌گیرد و لازم نیست بقیه نام دستور را تایپ کنید. این امکان علاوه بر اینکه سبب می‌شود سرعت تایپ برنامه افزایش یابد، باعث می‌شود دستورات با املای صحیح و بدون خطا نوشته شود.

۱- Intelligent Sense

۲- Space Bar

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



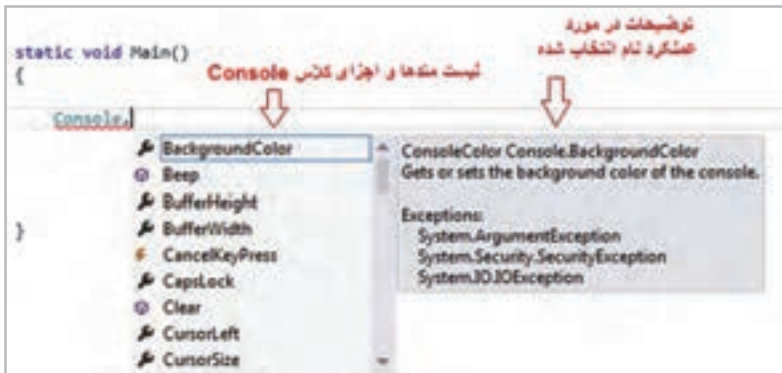
شکل ۷-۱- پنجره کدنویسی

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

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

نکته

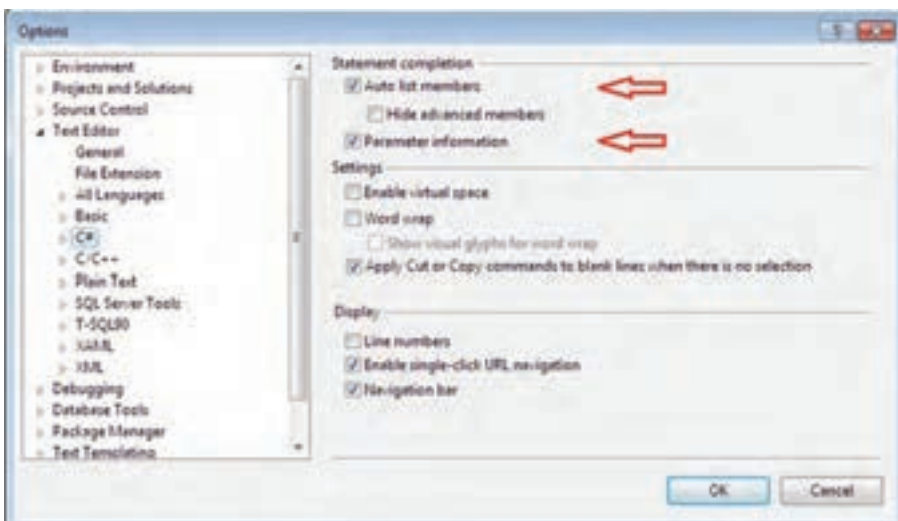
لیست مربوط به IntelliSense را در داخل متن برنامه، می توانید با زدن کلید ترکیبی CTRL+SPACE ظاهر کنید.



شکل ۸-۱- منوی IntelliSense

نکته

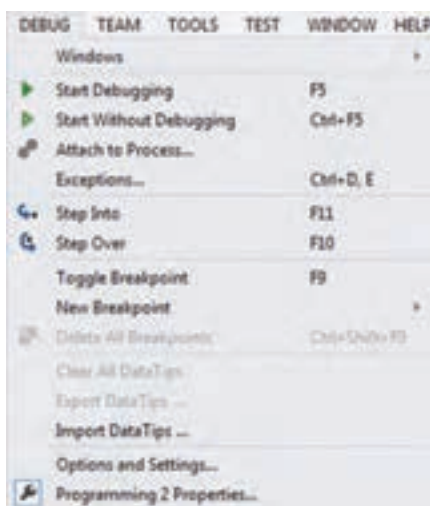
اگر امکان IntelliSense فعال نیست و از آن نمی توانید استفاده کنید، باید از طریق منوی Tools گزینه Options وارد قسمت Text Editor شوید و سپس در بخش C#, گزینه `Auto list members` و `Parameter information` را فعال کنید.



شکل ۹-۱- فعال یا غیر فعال کردن امکان IntelliSense

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

سؤال: تفاوت این دو گزینه در اجرای برنامه چیست؟



شکل ۱۰-۱- منوی Debug برای اجرای برنامه

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



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

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

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

۱-۳-۱-۱ دستورات ورودی و خروجی

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

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

۱-۳-۲-۱ تبدیل نوع داده: به خاطر دارید که برای نگهداری داده‌ها در برنامه، از متغیرها استفاده می‌شود. هر متغیر مانند ظرفی است که می‌تواند مقداری را در خود نگهداری کند. اندازه و گنجایش هر متغیر توسط نوع داده معین می‌شود. انواع مختلفی از داده‌ها در زبان `C#` وجود دارند که برای اعداد صحیح، اعداد اعشاری، حروف و کلمات مورد استفاده قرار می‌گیرند. مانند: `int`, `double`, `char`, `string`.

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

کار در کارگاه ۱

مثال ۱-۲: برنامه‌ای بنویسید که سن شما را بر حسب سال نشان دهد.

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

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

برنامه ۱-۲-۱- نمایش سن

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

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

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

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

برنامه ۳-۱- نمایش سن سال جاری

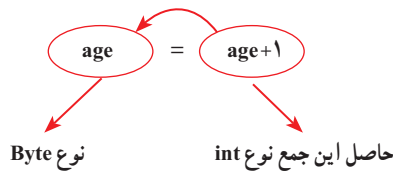
اما با دقت در متن برنامه متوجه می‌شویم که مترجم از دستور افزایش مقدار متغیر به اندازه یک واحد خطا می‌گیرد. اگر علامت ماوس را روی خط قرمز رنگ ببریم توضیح خطا به صورت زیر نوشته می‌شود:

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

شکل ۱۲-۱- خطای صادر شده

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

اکنون می‌خواهیم علت بروز مشکل در دستور: $age = age + 1$ در برنامه بالا را بررسی کنیم. عملگر + دارای دو عملوند است. یکی از عملوندها، متغیر age از نوع `byte` و عملوند دیگر عدد ۱ که عددی صحیح است و مترجم به طور پیش فرض اعداد صحیح را از نوع `int` در نظر می‌گیرد. ظرفیت داده `int`، چهار بایت و ظرفیت داده `byte` یک بایت است و نمی‌توان محتوای `int` را به طور مستقیم درون داده‌های کوچک‌تر از خودش جای داد. بنابراین حاصل عمل جمع، عددی از نوع صحیح `int` خواهد بود و هنگامی که بخواهید یک مقدار `int` را در یک متغیر کوچک از نوع `byte` قرار دهید این مقدار بزرگ‌تر از گنجایش ظرف مورد نظر شما است و بنابراین مترجم خطا اعلام می‌کند.



برای برطرف کردن اشکال، می‌توانید از مترجم بخواهید که حاصل جمع را به نوع `byte` تبدیل کند تا در متغیر age جای بگیرد. برای این منظور از تبدیل نوع استفاده می‌کنیم (برنامه ۴-۱). شکل کلی تبدیل نوع داده چنین است:

عبارت مبدأ (نوع داده مقصد)

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

برنامه ۴-۱- نوع دیگری از برنامه نمایش سن سال جاری

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

مثال ۳-۱: برنامه‌ای بنویسید که با استفاده از تبدیل نوع، عدد را به صورت کاراکتر و همچنین کاراکتر را به صورت عدد تبدیل کرده و نمایش دهد.

الگوریتم یا روش انجام کار: در این برنامه، در متد () Write محتوای متغیر aNumber را که عدد صحیح 67 قرار داده‌ایم، با استفاده از تبدیل نوع (char)، به عنوان کد اسکی یک کاراکتر در نظر گرفته می‌شود که در این صورت، معادل با حرف انگلیسی بزرگ C است. همچنین محتوای متغیر کاراکتری ch و کاراکتر 'C' با استفاده از تبدیل نوع (int) به یک عدد صحیح که همان کد کاراکتر است تبدیل شده، و با استفاده از متد () WriteLine نمایش داده می‌شوند.

```
using System;
class Review
{

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

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

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

برنامه ۵-۱- نمایش کدهای اسکی کاراکترهای C و #

مثال ۴-۱: برنامه‌ای بنویسید که نام کاربر را سؤال نماید و یک پیام خوشامدگویی به وی اعلام کند. الگوریتم یا روش انجام کار: در این برنامه برای دریافت نام کاربر، از متد `ReadLine()` استفاده می‌کنیم و نام کاربر را در یک متغیر ذخیره می‌نماییم. نوع متغیر باید از نوع رشته‌ای `String` باشد که بتوان یک رشته یا نام کاربر را در آن نگهداری کرد. سپس به وسیله متد `WriteLine()` نام کاربر به همراه پیام خوشامدگویی را در خروجی نمایش می‌دهیم. به برنامه زیر دقت کنید.

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

برنامه ۴-۱- خوشامدگویی به کاربر

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

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

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



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

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

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

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

{الگوی نمایش : عدد تراز، شماره}

اگر عدد تراز منفی باشد، مقدار موردنظر در فضای اختصاص یافته چپ چین می‌شود و اگر عدد تراز مثبت باشد، مقدار موردنظر در فضای اختصاص یافته راست چین می‌شود. الگوی نمایش از تعدادی کاراکتر تشکیل می‌شود که هر یک نحوه نمایش داده را مشخص می‌کند. مثلاً برای نمایش اعداد اعشاری با دو رقم اعشار (مثل میانگین نمره) از کاراکتر F استفاده می‌شود. در پیوست ۲، لیست کامل کاراکترهای الگوی نمایش برای انواع داده‌ها وجود دارد.

مثال ۵-۱ : در برنامه ۷-۱ میانگین سه نمره محاسبه شده است و سپس به دو صورت معمولی و همچنین با دو رقم اعشار نشان داده شده است. در هر دو تراز راست با فضای ۱۰ کاراکتر در نظر گرفته شده است.

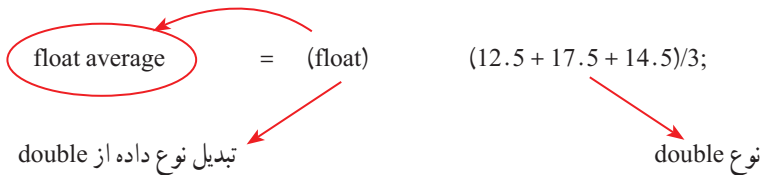

```

using System;
class Program
{

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

```

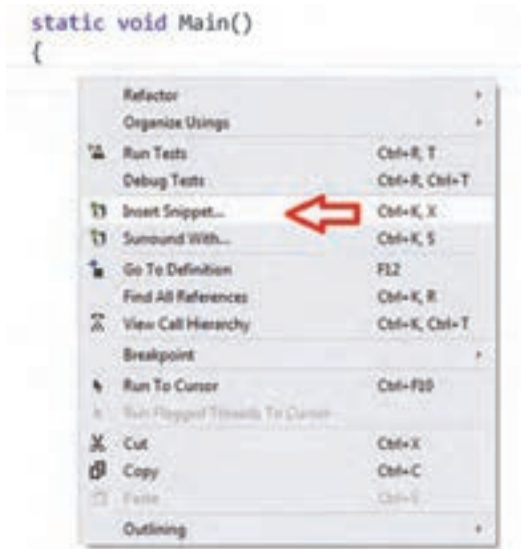
برنامه ۷-۱- نمایش عدد اعشاری با دو رقم اعشار و با تراز راست



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

۴-۳-۱ یادآوری دستورات شرطی

۴-۳-۱-۱ دستور if-else: از دستور if-else برای تصمیم‌گیری در اجرای دستورات استفاده می‌شود. در جلوی دستور if یک عبارت منطقی ساده و یا مرکب به عنوان شرط نوشته می‌شود، در صورت درست بودن شرط، دستور یا دستورات بعد از این عبارت اجرا می‌شود، در غیر این صورت دستور یا دستورات پس از else اجرا می‌شود. زبان‌های برنامه‌نویسی، امکاناتی را فراهم کرده‌اند که نیاز به حفظ کردن تمامی جزئیات و تاپ دستورات نیست، به عنوان مثال می‌توانید در پنجره کد نویسی کلیک راست کرده و از منویی که ظاهر می‌شود گزینه Insert Snippet را انتخاب کنید (شکل ۱۳-۱).



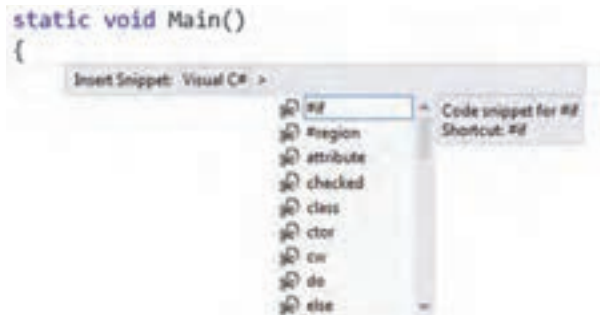
شکل ۱۳-۱- گزینه Insert Snippet



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

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

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



شکل ۱۵-۱- انتخاب دستورات برنامه نویسی در گزینه انتخاب شده

به عنوان مثال اگر بخواهیم دستور if را در این لیست پیدا کنیم، حرف i را تایپ می‌کنیم تا به سرعت if در لیست یافت شود. اکنون می‌توانید روی آن کلیک کنید (شکل ۱۵-۱). بعد از ظاهر شدن دستور انتخابی، می‌توانیم عبارت داخل پرانتز را به دلخواه تغییر دهیم (شکل ۱۶-۱).

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

شکل ۱۶-۱ ساختار دستور if

کار در کارگاه ۲

مثال ۱۶-۱: برنامه‌ای بنویسید که نمره یک درس را از ورودی دریافت کند و تشخیص دهد آیا این نمره معتبر است یا خیر.

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

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

```

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

```

برنامه ۸-۱- معتبر بودن نمره

مثال ۷-۱: مثال قبل را توسعه دهید به طوری که قبولی یا مردودی را نیز معین کند. الگوریتم یا روش انجام کار: برای تشخیص قبولی یا مردودی، باید یک دستور if به برنامه قبل اضافه کنیم. اگر نمره کمتر از ۱۰ باشد پیام "شما نمره قبولی نیاوردید. تلاش بیشتری کنید." بر روی صفحه نشان داده شود و در غیر این صورت (در قسمت else آن)، پیام "شما قبول شدید." نمایش داده شود.

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

```

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

```

سؤال: دستور if جدید را در چه قسمتی از برنامه ۸-۱ درج کنیم؟

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

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

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

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

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

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

۲-۴-۱- عملگر سه تایی یا علامت سؤال : دستور if را در قطعه برنامه زیر در نظر

بگیرید :

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

دستورات بالا را می توان به صورت خلاصه با استفاده از عملگر علامت سؤال به صورت زیر

نوشت :

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

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

مقدار دوم : مقدار اول ؟ (عبارت منطقی)

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

سؤال: به نظر شما چرا عملگر ؟ عملگر سه تایی نامیده می شود؟

مثال ۸-۱: در برنامه ۹-۱ به جای دستور if از عملگر سه تایی استفاده کنید.

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

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

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

برنامه ۹-۱- تشخیص زوج و فرد بودن اعداد

۳-۴-۳-۱- دستور switch: برای مواقعی که بخواهیم حالت‌های مختلف یک عبارت را بررسی کرده و بر اساس آن دستورات را اجرا کنیم از دستور switch استفاده می‌کنیم.

کار در کارگاه ۳

مثال ۹-۱: برنامه ۱۰-۱ به وسیله یکی از هنرجویان نوشته شده است. کار این برنامه را بررسی

می‌کنیم:

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

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

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

```
using System;
class Program
{
    static void Main ( )
    {
        for (int i = 1; i <= 2; i++)
        {
            int n = int.Parse(Console.ReadLine ( ));
            switch (n)
            {
                case 1:
                    Console.BackgroundColor = ConsoleColor.Red;
                    Console.Clear();
                    break;
                case 2:
                    Console.BackgroundColor = ConsoleColor.Green;
                    Console.Clear();
                    break;
                case 3:
                    Console.BackgroundColor = ConsoleColor.Blue;
                    Console.Clear();
                    break;
                default:
                    Console.WriteLine ("ERROR");
                    break;
            }
        }
    }
}
```

برنامه ۱۰-۱- تغییر رنگ صفحه کنسول

۵-۳-۱- یادآوری دستورات تکرار یا حلقه

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

۵-۳-۱- دستور `while`: برای مواقعی که یک یا چند دستور باید تا زمانی که حاصل یک عبارت منطقی درست است اجرا شوند، از دستور `while` استفاده می‌شود.

کار در کارگاه ۴

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

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

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

```
using System;
```

```
class Review
```

```
{
```

```
    static void Main ( )
```

```
    {
```

```
        int number;
```

```
        string input;
```

```
        Console. WriteLine ( " Counting down to one" );
```

```
        Console. Write ( " Enter a number (1-100): " );
```

```
        input = Console. ReadLine ( );
```

```
        number = int. Parse (input);
```

```
        while (number > 0)
```

```

    {
        Console.WriteLine (number);
        number --;
    }
}

```

برنامه ۱۱-۱- نمایش اعداد متوالی به صورت کاهشی تا صفر

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

در قطعه برنامه زیر به دستورات تکرار شونده توجه کنید :

```

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

```

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

```

while (number > 0)
    Console.WriteLine (number --);

```

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

مثال ۱۱-۱: برنامه‌ای بنویسید که یک عدد را از کاربر دریافت کند و تعداد ارقام آن را شمارش و نشان دهد.

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

عدد، به دو روش زیر می توانیم عمل کنیم :

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

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

```
using System;
```

```
class Review
```

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

برنامه ۱۲-۱- نمایش شمارش ارقام

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

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

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

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

مثال ۱-۱۳ : برنامه‌ای بنویسید که نمرات درس برنامه‌سازی ۲ مربوط به یک کلاس ۱۰ نفری را دریافت کند و مجموع و میانگین نمرات را حساب کند.

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

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

برنامه ۱-۱۳- محاسبه مجموع و معدل ۱۰نمره

۱-۴- استفاده از مقدار ثابت^۱ در برنامه

مثال ۱-۴: برنامه مثال قبل را برای یک کلاس با ۱۵ نفر دانش آموز تغییر (توسعه) دهید.

الگوریتم یا روش انجام کار: در این حالت لازم است به جای دریافت ۱۰ عدد، پانزده عدد دریافت کنیم، بنابراین حلقه تکرار for باید به تعداد ۱۵ بار اجرا شود. برای این منظور کافی است عدد ۱۰ را در حلقه for را به ۱۵ تغییر می دهیم. همچنین باید مجموع نمرات را به تعداد نمرات دریافتی یعنی عدد ۱۵ تقسیم کنیم. به این ترتیب باید در انتهای برنامه مجموع نمرات (total) را به عدد ۱۵ تقسیم کنیم. اما فرض کنید که برنامه نویس فراموش کند عدد ۱۰ دوم را تغییر دهد در این صورت با اجرای برنامه پانزده نمره دریافت می شود و مجموع نمرات به درستی حساب می شود اما میانگین نمرات به اشتباه حساب می شود. برای جلوگیری از چنین اشتباهاتی و همچنین توسعه راحت برنامه می توان در ابتدای کلاس یک شناسه به عنوان یک ثابت تعریف کرد و عدد ۱۰ را به آن نسبت داد سپس در سرتاسر برنامه هر جا که به عدد ۱۰ نیاز بود از آن شناسه استفاده کرد.

برای تعریف شناسه یا نام ثابت از کلمه کلیدی const استفاده می شود. معمولاً نامی که برای اعداد ثابت تعریف می شود با حروف بزرگ نوشته می شود تا در برنامه مشخص باشد که این شناسه، یک مقدار ثابت است. نحوه تعریف ثابت ها، مانند تعریف متغیرها می باشد با این تفاوت که در ابتدای تعریف آنها کلمه const قرار دارد.

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

برنامه ۱-۴ را در نظر بگیرید:

```
using System;
```

```
class Review
```

```
{
```

```
    const int SIZE = 15; ↔
```

```
    static void Main ()
```

```
{
```

```
    float number, total =0, average;
```

```
    string input;
```

```
    for (int i = 0; i < SIZE; i++) ↔
```

```
{
```

^۱_ Constant

```

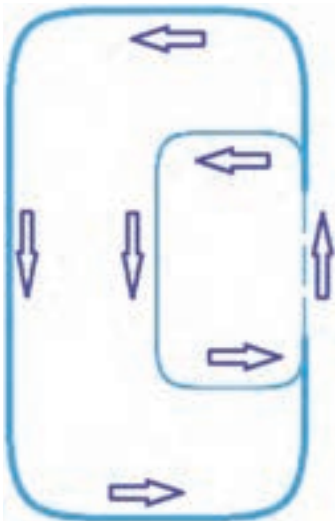
Console. Write ( " Enter a mark: " );
input = Console. ReadLine ( );
number = float. Parse (input);
total = total + number;
}
average = total / SIZE ; ←
Console. WriteLine ( "Total : " + total);
Console. WriteLine ( "Average : " + average);
}
}

```

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

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

۵-۱- حلقه های متداخل (تو در تو)



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

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

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

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

قرار دارد و بنابراین حلقه تودرتو را تشکیل می دهند.

الگوریتم یا روش انجام کار: در این برنامه، دستور for با متغیر شمارنده i، حلقه بیرونی^۱ و

دستور for با متغیر شمارنده j، حلقه داخلی^۲ را تشکیل می دهد. کامپیوتر به ازای هر بار اجرای حلقه

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

```
using System;
```

```
class Review
```

```
{
```

```
    static void Main ( )
```

حلقه بیرونی

```
    {
```

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

```
        {
```

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

حلقه داخلی

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

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

```
        }
```

```
    }
```

```
}
```

برنامه ۱۵-۱ حلقه تو در تو

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

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

```
*****
```

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

```
*****
```

```
*****
```

```
*****
```

```
*****
```

۱_ Outer Loop

۲_ Inner Loop

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

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

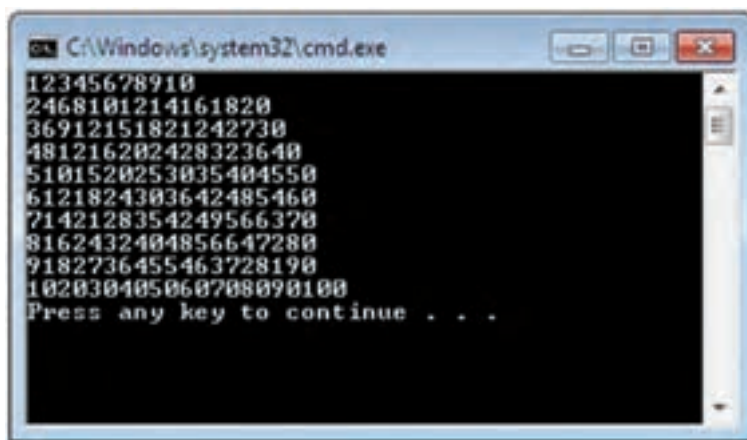
کار در کارگاه ۵

مثال ۱۶-۱: برنامه‌ای بنویسید که جدول ضرب اعداد ۱ تا ۱۰، را روی صفحه نشان دهد. الگوریتم یا روش انجام کار: برای نمایش جدول ضرب اعداد ۱ تا ۱۰، کافی است در برنامه قبل به جای نمایش رشته "*" حاصل عبارت $i * j$ را محاسبه و روی صفحه چاپ کنیم.

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

برنامه ۱۶-۱- جدول ضرب اعداد ۱ تا ۱۰

نتیجه اجرا یا خروجی برنامه فوق چنین خواهد بود :



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

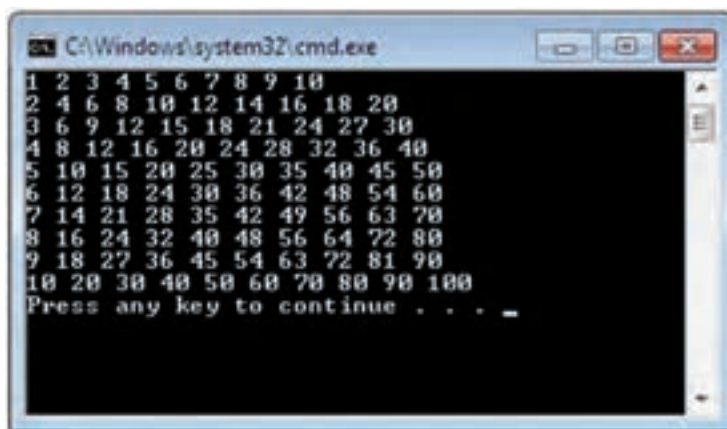
شکل ۱۸-۱- خروجی برنامه جدول ضرب

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

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



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



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

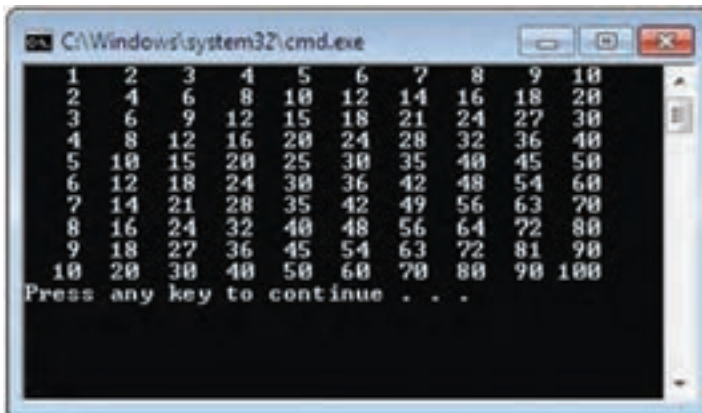
شکل ۱۹-۱- جدا کردن اعداد جدول ضرب

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

خوشبختانه در زبان C#، راه حل چنین مشکلی در نظر گرفته شده است و امکان شکل یا الگوی^۱ نمایش از قبل پیش بینی شده است. اعداد جدول ضرب دو رقمی هستند به جز حالت خاص ضرب ۱۰ در ۱۰ است که حاصل برابر ۱۰۰ و یک عدد ۳ رقمی است بنابراین اگر برای چاپ هر عدد ۳ مکان و یک مکان نیز به عنوان فاصله در نظر بگیریم، در مجموع فضایی به اندازه ۴ کاراکتر، حداقل فضای لازم برای جداسازی اعداد است. در علامت {0} بعد از عدد صفر، تعداد فضای اختصاص داده شده را به صورت زیر مشخص می‌کنیم:

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

در این صورت خروجی برنامه چنین خواهد بود:



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

شکل ۲۰-۱. خروجی نهایی برنامه جدول ضرب

۱-۶- کاراکتر ۲


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

۱- Format String

۲- Back Slash Character

به این مجموعه کاراکترها که کاراکتر اول \ و دومی کاراکتر دیگری است، دنباله فرار^۱ گفته می‌شود. یکی از کاربردهای این دنباله، نمایش علامت" در یک رشته است. اگر بخواهیم جمله‌ای را همراه با علامت" روی صفحه نمایش چاپ کنیم به مشکل برخورد می‌کنیم. چون معنای معمول این کاراکتر، نشانه شروع و پایان یک رشته است و در وسط یک رشته نمی‌تواند قرار گیرد. لذا باید معنی معمول آن را تغییر دهیم. بنابراین از دنباله \" برای بیان این کاراکتر استفاده می‌کنیم. :

Console. WriteLine ("C# is pronounced \"see Sharp\" ");



؟ سؤال: خروجی دستور بالا چیست؟

در جدول زیر تعدادی از دنباله‌های معنی‌دار نشان داده شده‌اند.

جدول ۱-۱- دنباله معنی‌دار

دنباله	عملکرد
\a	ایجاد یک بوق هشدار ^۲
\b	حذف یک کاراکتر (Backspace)
\n	ایجاد یک خط خالی (New Line)
\t	ایجاد یک فاصله افقی زیاد tab
\'	ایجاد یک تک کوتیشن (')
\"	ایجاد یک دابل کوتیشن (")
\\	ایجاد یک Back slash (\)

نکته

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

می‌شود.

۱- Escape Sequence

۲- Alarm

مثال ۱-۱۷: در برنامه ۱-۱۷ دنباله‌های معنی‌دار استفاده شده‌اند. خروجی هر دستور به صورت توضیح در جلوی آن نشان داده شده است.

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

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

برنامه ۱-۱۷- نمایش دنباله معنی‌دار

خودآزمایی فصل اول

الف) درستی یا نادرستی هر عبارت را تعیین کنید.

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

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

—	—	—	—
—	—	—	—
—	—	—	—
—	—	—	—

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

۱۱- عملکرد هر یک از سه دستور زیر را بررسی کنید. آیا خروجی این سه دستور یکسان است؟
(الف)

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

(ب)

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

(ج)

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



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

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

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

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

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

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

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

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


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

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

طرح (۱)

طرح (۲)

طرح (۳)

طرح (۴)

۵- خروجی اجرای دستورات زیر چیست؟

```
string columns = "Column 1\tColumn 2\tColumn 3";
```

```
string rows = "Row 1\r\nRow 2\r\nRow 3";
```

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

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

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

Lucky Numbers:

a) Write a program to find and print all four-digit numbers of the type abcd,

where: $a+b = c+d$

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

(Hint: Use four nested for-loops – one for each digit as seen below)

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

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

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

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

واژگان و اصطلاحات انگلیسی فصل اول

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

آرایه

بخاطر دارید که متغیر مکانی از حافظه است که مانند یک ظرف مقداری را در خود نگه می‌دارد و با ورود مقدار جدید، مقدار قبلی آن از بین می‌رود. برخی مواقع در برنامه‌ها نیاز به نگه‌داری داده‌ها داریم و گاهی مقدار این داده‌ها خیلی زیادند و نمی‌توانیم ده‌ها و یا حتی صدها متغیر تعریف کنیم. بنابراین لیستی از داده‌های همنام تعریف می‌کنیم. در این فصل می‌خواهیم از مفهوم آرایه^۱ برای تعریف چنین متغیرهایی استفاده کنیم و اهمیت و کاربرد آنها را با چند مثال مورد توجه قرار دهیم.

پس از پایان این فصل انتظار می‌رود که فراگیر بتواند:

- ۱- آرایه و کاربرد آن را توضیح دهد.
- ۲- متغیری از نوع آرایه تعریف کند و آن را در برنامه به کار بندد.
- ۳- روش مرتب‌سازی حبابی را شرح داده و برنامه مرتب‌سازی عناصر آرایه را بنویسد.
- ۴- با استفاده از دستور `foreach` به عناصر آرایه و رشته دسترسی پیدا کند.
- ۵- روش‌های جست‌وجو را توضیح داده و تفاوت آنها را بیان نماید.
- ۶- کلید جست‌وجو را با استفاده از برنامه جستجوی خطی و دودویی پیدا کند.

^۱- Array

۱-۲- تعریف آرایه

به مثال زیر توجه کنید.

مثال ۱-۲: فرض کنید می‌خواهیم برنامه‌ای بنویسیم که در یک کلاس ۱۶ نفری محاسبه کند چند درصد نمره‌ها، بالاتر از میانگین کلاس در درس برنامه‌سازی ۲ هستند.

الگوریتم یا روش انجام کار: برای نوشتن این برنامه، پس از دریافت نمرات کل کلاس لازم است ابتدا مجموع و میانگین نمرات را محاسبه کنیم. سپس هر نمره را با میانگین، مقایسه کرده و محاسبه کنیم چه تعدادی از نمرات بیشتر از میانگین کلاس بوده‌اند. برای شمردن این نمرات از شمارنده استفاده می‌کنیم و سپس درصد می‌گیریم.

برای نوشتن این برنامه، ناگزیریم از نمرات کلاس دوبار استفاده کنیم:

– بار اول در خواندن ورودی‌ها.

– بار دوم مقایسه ورودی‌ها با میانگین محاسبه شده.

طبق روال کتاب برنامه‌سازی ۱ متغیری را از نوع اعشاری تعریف می‌کنیم و در یک حلقه for،

۱۶ بار دریافت کرده و جمع و میانگین را محاسبه می‌کنیم.

سؤال: آیا می‌توانیم برای بار دوم از داده‌های ورودی استفاده کنیم و عمل مقایسه را

انجام دهیم؟

این نیاز ما به داده‌های ورودی سبب می‌شود به جای تعریف یک متغیر اعشاری برای نمره و ۱۶ بار تکرار دریافت داده، از ۱۶ متغیر اعشاری که یک لیست پشت سر هم را تشکیل می‌دهند استفاده کنیم، تا در صورت نیاز به دفعات لازم از آنها بهره ببریم.

در مبحث حلقه‌ها یاد گرفتیم که برای دریافت ۱۶ نمره به صورت زیر عمل کنیم:

```
string input;
float mark, total = 0;
for (int i = 0; i <= 15; i++)
{
    Console. Write ("Enter a mark: ");
    input = Console. ReadLine ();
    mark = float. Parse (input);
    total = total + mark;
}
```

سؤال: متغیر mark در این روش پس از پایان حلقه، چه مقداری را در خود دارد؟

در روش بالا ما به همه ۱۶ داده‌ای که درون یک متغیر ریخته شده است دسترسی نداریم. چرا؟ بنابراین چگونه می‌توانیم برای بار دوم به داده‌ها دسترسی پیدا کنیم؟ پاسخ این سؤال در تعریف متغیر قرار دارد. ما نیازمند خانه‌های بیشتری از حافظه هستیم تا داده‌ها را به صورت تک تک در آنها قرار دهیم. این خانه‌های پشت سرهم و همنام آرایه نام دارند.

تعریف آرایه: مکان‌های متوالی در حافظه کامپیوتر که در آن داده‌هایی از یک نوع نگهداری می‌شوند آرایه نام دارند. مثلاً ۱۶ مکان متوالی را در نظر بگیرید که در هر یک از آنها نمره درسی دانش‌آموزان یک کلاس قرار دارد.

17.5	15	19	16.5	.	.	.	20	14	18
------	----	----	------	---	---	---	----	----	----

هر یک از مکان‌ها، یک عنصر^۱ آرایه نامیده می‌شود. برای تفکیک و دسترسی به این مکان‌ها، از یک عدد صحیح به نام اندیس^۲ استفاده می‌شود. (مانند پلاک خانه‌های یک کوچه) اولین عنصر آرایه با اندیس صفر مشخص می‌شود و عنصر بعد از آن شماره^۳ یک و به ترتیب جلو می‌رود تا آخرین عنصر که یک واحد کمتر از تعداد عناصر آرایه است. بنابراین آرایه‌ای با ۵ عنصر دارای اندیس‌های ۰، ۱، ۲، ۳، ۴، ۵ می‌باشد. تعداد عناصر آرایه را اندازه یا طول^۳ آرایه می‌نامند.

سؤال: در مثال بالا اندیس آخرین عنصر آرایه چه عددی است؟

۲-۲- نحوه ایجاد یا تعریف آرایه در زبان C#

به خاطر دارید که برای تعریف یک متغیر از نوع float به شکل زیر عمل می‌کردیم:

```
float mark;
```

برای تعریف یک آرایه از نوع float مشابه تعریف یک متغیر به شکل زیر عمل می‌شود:

```
float[] mark;
```

۱- Element

۲- Index

۳- Length Or Size

اما تعریف بالا هنوز کامل نیست و فقط مرحله اول تعریف است. در زبان سی شارپ برای تعریف یک آرایه باید دو مرحله‌ای عمل کنیم :

شکل کلی مرحله اول تعریف آرایه :

؛ نام متغیر آرایه [] نوع داده

در تعریف متغیر گفته شد مکانی از حافظه برای نگهداری داده‌هاست اما با تعریف بالا هنوز مکانی از حافظه برای آرایه تخصیص نیافته است. با استفاده از عملگر `new` و مشخص کردن اندازه آرایه در مرحله دوم، تعریف کامل می‌شود.

شکل کلی مرحله دوم تعریف آرایه :

؛ [اندازه آرایه] نوع داده `new`

بنابراین برای تعریف متغیر `mark` با ۲۰ عنصر به صورت زیر عمل می‌کنیم :

`float[] mark;` ← مرحله اول

`new float [20]` ← مرحله دوم

به جای دو دستور بالا می‌توانیم دستور زیر را جایگزین نماییم :

`float[] list = new float [20];`

شکل کلی تعریف آرایه :

[اندازه آرایه] نوع داده `new` = نام آرایه [] نوع داده

۳-۲- دسترسی، مقداردهی و نمایش عناصر آرایه

۱-۳-۲- دسترسی به عناصر آرایه : برای دسترسی به عناصر آرایه، از نام آرایه با

ذکراندیس به صورت زیر استفاده می‌کنیم :

؛ [اندیس] نام متغیر آرایه

برای مثال `list[0]` بیانگر اولین عنصر آرایه و `list[19]` آخرین عنصر، که بیستمین عضو آرایه `list` است.

۲-۳-۲ مقداردهی عناصر آرایه : برای مقداردهی عناصر آرایه به چند روش می‌توان عمل کرد. یک روش مقداردهی، با استفاده از دستور انتساب است. برای مثال ذخیره کردن نمره 17.5 در اولین عنصر آرایه را با استفاده از دستور انتساب مشاهده می‌کنید.

```
list[0]=17.5f;
```

سؤال! به یاد دارید که چرا در انتهای عدد 17.5 حرف `f` قرار دارد؟

برای مقداردهی هر عنصر آرایه در طول اجرای برنامه، می‌توانید داده‌ای را از کاربر دریافت کرده و آن را در عنصری از آرایه نگهداری کنید. مثلاً نمره‌ای را از کاربر دریافت و در سومین عنصر ذخیره می‌کنیم :

```
string input;  
float [] list = new float [20];  
Console. Write ("Enter your mark: ");  
input = Console. ReadLine();  
list[2] = float. Parse (input); ← عنصر سوم
```

و یا برای دریافت تمام نمرات و قراردادن آنها در آرایه `list` به صورت زیر عمل می‌کنیم :

```
for (int i = 0; i < 20; i++)  
{  
    Console. Write ("Enter a mark: ");  
    input = Console. ReadLine ();  
    list [i] = float. Parse (input);  
}
```

روش دیگر مقداردهی عناصر آرایه در هنگام تعریف و ایجاد آرایه است که در انتهای تعریف آرایه در بین علامت‌های { } مقدار هر عنصر از آرایه را به ترتیب معین می‌کنیم :

; { مقدار، ...، مقدار، مقدار } [] نوع داده new = نام آرایه [] نوع داده

همان‌طور که مشاهده می‌کنید عددی بین علامت‌های [] وجود ندارد، بنابراین اندازه آرایه را ذکر نمی‌کنیم بلکه اندازه چنین آرایه‌هایی با تعداد مقادیر نوشته شده بین علامت‌های { } تعیین می‌شود. مثلاً برای ایجاد آرایه‌ای برای سکه‌های رایج بر حسب ریال، به صورت زیر تعریف می‌کنیم.

```
int [] coin = new int [] {500,1000,2000,5000};
```

دستور بالا آرایه‌ای به نام coin شامل چهار عنصر ایجاد می‌کند که هر عنصر آن مقدار یک سکه را مشخص می‌کند که از سکه ۵۰۰ ریالی شروع و به سکه ۵۰۰۰ ریالی ختم می‌شود. دستور بالا را می‌توان با دستورات زیر جایگزین نمود :

```
int[] coin = new int [4];
```

```
coin [0] = 500;
```

```
coin [1] = 1000;
```

```
coin [2] = 2000;
```

```
coin [3] = 5000;
```

همچنین با توجه به اینکه آرایه coin دارای مقادیر اولیه مشخص است می‌توان بدون استفاده از عملگر new آن را ایجاد کرد. بنابراین دستورات بالا را می‌توان با دستور زیر جایگزین نمود :

```
int[] coin = {500, 1000, 2000, 5000};
```

نکته

بعد از ایجاد آرایه، نمی‌توانید اندازه آن را تغییر دهید یعنی نمی‌توانید عنصری به آن اضافه و یا کم کنید.

بنابراین می‌توان در تعریف آرایه‌هایی که از قبل مقادیر اولیه‌شان مشخص است بدون استفاده از عملگر new به شکل زیر عمل کرد :

{ مقدار، ...، مقدار، مقدار } = نام آرایه [] نوع داده

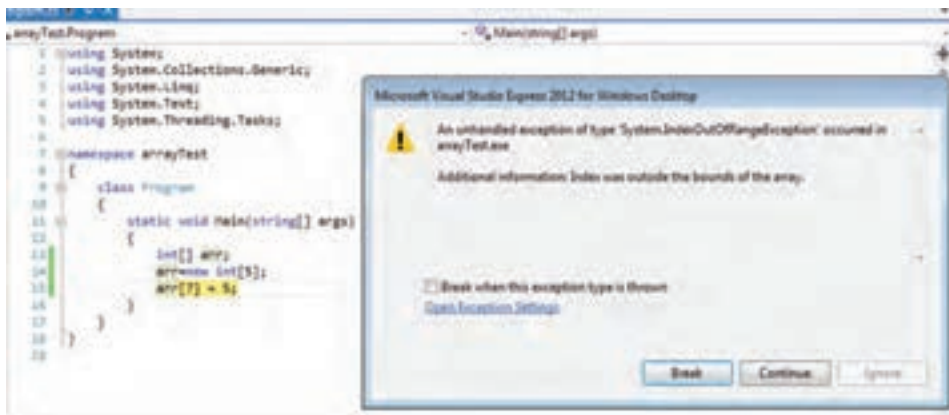
۳-۲- نمایش عناصر آرایه : برای نمایش محتوای عناصر آرایه می‌توان از متد Write() یا WriteLine() استفاده نماییم. مثلاً برای نمایش محتوای آخرین عنصر آرایه نمرات، خواهیم داشت :

System.Console.WriteLine("Last mark: " + list [19]);

سؤال؟ عناصر اندیس‌های زوج این آرایه را با حلقه for نمایش دهید؟

نکته

در زبان C# محدوده اندیس آرایه کنترل می‌شود و نباید از عدد صفر کمتر و همچنین از اندازه آرایه بیشتر یا مساوی باشد. اگر برنامه‌نویس اشتباه کند و اندیس بالاتری را استفاده کند در هنگام ترجمه برنامه با خطا روبه‌رو می‌شود (شکل ۲-۱).



شکل ۲-۱- خطای سرریزی محدوده اندیس آرایه

در آغاز فصل، مثال ۱-۲ را با الگوریتم ذکر کردیم. اکنون می‌خواهیم مرور دوباره داشته باشیم و برنامه را در محیط VS بنویسیم. می‌خواستیم برنامه‌ای بنویسیم که در یک کلاس ۱۶ نفری محاسبه کند چند درصد نمره‌ها، بالاتر از میانگین کلاس در درس برنامه‌سازی ۲ هستند.

الگوریتم یا روش انجام کار: برای نوشتن این برنامه، پس از دریافت نمرات به وسیله حلقه در آرایه و محاسبه میانگین، در حلقه دوم، عمل مقایسه صورت می‌گیرد. هر نمره با میانگین مقایسه شده و در صورتی که نمره بیشتر از میانگین کلاس باشد، به وسیله یک متغیر به عنوان شمارنده، تعداد را محاسبه می‌کنیم و سپس درصد می‌گیریم.

```
using System;
```

```
class Program
```

```
{
    static void Main(string[] args)
    {
        float[] mark;
        mark = new float[15];
        float avg, percen, sum = 0;
        byte counter = 0;

        for (int i = 0; i <= 15; i++)
        {
            Console.WriteLine("enter the {0}th mark: ", i + 1);
            mark[i] = float.Parse(Console.ReadLine());
            sum = sum + mark[i];
        }
        avg = sum / 16;
        for (int i = 0; i <= 15; i++)
            if (mark[i] > avg)
                counter++;
        Console.WriteLine(" {0} person(s) > AVG ", counter);
        Console.WriteLine("-----");
    }
}
```

```

Console.WriteLine();
percen = (counter / 16f) * 100
Console.WriteLine(" {0} % > AVG ",percen);
Console.ReadKey();

}
}

```

برنامه ۱-۲- محاسبه درصد نمرات بالاتر از میانگین کلاس

۲-۴- حلقه foreach

برای مواقعی که بخواهیم محتوای تمام عناصر آرایه را مورد استفاده قرار دهیم یا آنها را روی صفحه نمایش ببینیم، می توان به جای حلقه for از حلقه foreach استفاده کنیم. دستور foreach روی هر عنصر از داده قابل شمارش تکرار می شود. مثلاً هر دو نوع array و string قابل شمارش هستند.

شکل کلی این دستور به صورت زیر است :

(نام آرایه in نام متغیر نوع داده) foreach

؛ دستور

دستور زیر عناصر آرایه score را روی صفحه نمایش می دهد.

```

foreach (int aValue in score)
    Console.WriteLine(aValue);

```

همان طور که مشاهده می کنید در هنگام دسترسی به عناصر آرایه از اندیس کمک نمی گیریم بلکه با هر بار تکرار حلقه، متغیر aValue نقش یک عنصر از آرایه را به عهده می گیرد که با کلمه کلیدی in به آرایه score متصل می شود.

مثال ۲-۲: برنامه‌ای بنویسید که نمرات دانش‌آموزان را دریافت کند و تعداد مردودی‌ها را نمایش دهد.

الگوریتم یا روش انجام کار: پس از تعریف آرایه و دریافت نمرات، توسط دستور `foreach` روی عناصر آرایه حرکت می‌کنیم و شرط مردودی که نمره کمتر از ۱۲ است را بررسی کرده و از متغیر شمارنده کمک می‌گیریم (برنامه ۲-۲).

```
using System;
```

```
class MarkOfSudents
{
    static void Main(string[] args)
    {
        byte counter = 0;
        float[] mark = new float[15];
        for (int i = 0; i < 15; i++)
        {
            Console.WriteLine("Enter the mark of {0}th student : ", i + 1);
            mark[i] = float.Parse(Console.ReadLine());
        }
        foreach (float m in mark)
            if (m < 12) counter++;
        Console.WriteLine("-----");
        Console.WriteLine(" the number of failed student(s) is : " + counter);
        Console.ReadKey();
    }
}
```

برنامه ۲-۲- دریافت نمرات و شمارش مردودی‌ها

مثال ۲-۳: رشته‌ای را از ورودی دریافت کنید و تک تک کاراکترهای آن را در خطوط جداگانه چاپ کنید.

الگوریتم یا روش انجام کار: با توجه به اینکه داده رشته‌ای نیز مانند آرایه از عناصر قابل شمارشی تشکیل شده است، می‌توان با استفاده از دستور `foreach` به تک تک عناصر آن دسترسی پیدا

کرد. بنابراین رشته‌ای را دریافت می‌کنیم و با استفاده از متغیر نوع کاراکتر به تک تک کاراکترهای رشته دست می‌یابیم (برنامه ۳-۲).

```
using System;

class Testforeach
{
    static void Main()
    {
        string name;
        Console.WriteLine("Enter a Name:");
        name = Console.ReadLine();
        foreach (char ch in name)
            Console.WriteLine(ch);
    }
}
```

برنامه ۳-۲ - نمایش کاراکترهای رشته در خطوط جداگانه

ویژگی *Length*

آرایه `names` را در نظر بگیرید.

```
string[] names = new string[10];
```

طول آرایه `names` برابر ۱۰ است. در زبان C# می‌توان توسط ویژگی `Length` از اندازه یک آرایه یا تعداد عناصر آن مطلع شد.

Length • نام آرایه

مثال ۴-۲: در قسمت آزمون چندگزینه‌ای یک برنامه آموزشی، پاسخ صحیح سؤالات در آرایه‌ای به نام `answer` قرار دارد، می‌خواهیم پاسخ صحیح سؤالات یا محتوای عناصر این آرایه را نمایش دهیم.

الگوریتم یا روش انجام کار: برای نمایش عناصر آرایه `answer` از حلقه `for` استفاده

می‌کنیم. شمارنده حلقه باید از صفر شروع و تا آخرین عنصر آرایه که یک واحد کمتر از تعداد عناصر آرایه است پیش رود. بنابراین به تعداد عناصر آرایه نیاز داریم که از ویژگی Length استفاده می‌کنیم:

↓

```
for (int i=0; i<answer. Length ; i++)  
    Console. Write (answer [i]);
```

قطعه برنامه ۴-۲- نمایش پاسخ‌های صحیح آرایه answer

سؤال: به نظر شما این ویژگی در چه برنامه‌هایی ممکن است مورد استفاده قرار بگیرد؟

۵-۲- تعیین اندازه آرایه در هنگام اجرای برنامه

در مثال‌هایی که تاکنون انجام دادیم اندازه یا تعداد داده‌ها از قبل مشخص بود و اندازه آرایه در متن برنامه مشخص شده بود، اما حالتی را در نظر بگیرید که تعداد داده‌هایی که قرار است کاربر وارد کند قبل از اجرای برنامه نامشخص باشد در این حالت چه باید کرد؟ مثلاً می‌خواهیم برنامه‌ای بنویسیم که میانگین نمرات دانش‌آموزان کلاس را محاسبه کند و برای هر کلاس، با تعداد دانش‌آموزان مختلف، قابل استفاده باشد.

روش اول: می‌توانیم یک آرایه با تعداد عناصر زیاد مثلاً ۵۰ عنصر را پیش‌بینی کنیم و داده‌ها را از کاربر دریافت و در آرایه قرار دهیم و پس از ورود آخرین نمره، کاربر یک عدد خاص مثلاً عدد ۱- را برای پایان دادن به ورودی، دریافت کند و در این صورت حلقه دریافت نمرات پایان یابد. در این روش با وارد کردن هر نمره نیز یک شمارنده را افزایش می‌دهیم تا تعداد نمرات را در اختیار داشته باشیم و در این صورت، ویژگی Length دیگر در حلقه کاربرد ندارد. چرا؟ در این روش همواره تعدادی از عناصر آرایه خالی و بدون استفاده هستند.

روش دوم: ابتدا تعداد نمرات را از کاربر سؤال کنیم و سپس آرایه‌ای به همان میزان ایجاد کرده و با استفاده از حلقه for مانند مثال‌های قبلی اقدام به دریافت نمرات کرده و از ویژگی Length نیز به عنوان شرط انتهای حلقه استفاده کنیم.

مثال ۵-۲: می‌خواهیم برنامه‌ای بنویسیم که میانگین نمرات دانش‌آموزان یک کلاس با هر تعداد دانش‌آموز را محاسبه کند.

الگوریتم یا روش انجام کار : برای نوشتن چنین برنامه‌ای از روش دوم که توضیح داده شد استفاده می‌کنیم. در برنامه زیر به نحوه تعریف آرایه list توجه کنید.

```
using System;

class StudentList
{
    static void Main()
    {
        string input;
        int listSize;
        Console. Write ("Enter number of the students: ");
        input = Console. ReadLine ();
        listSize = int. Parse (input);

        float [] list = new float [listSize]; ←
        float total = 0;

        for (int i = 0; i < list. Length ; i++)
        {
            Console. Write ("Enter a mark: ");
            input = Console. ReadLine();
            list[i] = float. Parse (input);
            total += list [i];
        }
        Console. WriteLine ("Average of the marks: {0}", total / list.
Length);
    }
}
```

برنامه ۵-۲- محاسبه میانگین نمرات با تعداد نامشخص

۶-۲- مرتب کردن داده‌های یک لیست

الگوریتم‌های مختلفی برای مرتب کردن عناصر لیست وجود دارد بعضی از آنها خیلی ساده و ابتدایی هستند مانند روش مرتب‌سازی حبابی^۱ که زمان زیادی طول می‌کشد تا لیست مرتب شود و برخی از روش‌ها وجود دارند مانند روش مرتب‌سازی سریع^۲ که سرعت بسیار بالایی دارد.

مثال ۶-۲: نمرات دانش‌آموزان یک کلاس را دریافت نموده و به صورت صعودی مرتب کنید. سپس رتبه‌های اول تا سوم را اعلام کنید.

الگوریتم یا روش انجام کار: فرض کنید می‌خواهیم لیستی از نمرات دانش‌آموزان را که به صورت زیر است مرتب کنیم:

۱۷, ۱۴, ۸/۵, ۱۳, ۲°

بعد از مرتب کردن، لیست به صورت زیر خواهد شد:

۸/۵, ۱۳, ۱۴, ۱۷, ۲°

در این روش از ابتدای لیست شروع کرده و دو عنصر اول و دوم را با یکدیگر مقایسه می‌کنیم. اگر ترتیب آنها درست نبود، آنها را جابه‌جا می‌کنیم. سپس به سراغ عنصر دوم و سوم می‌رویم و عمل مقایسه و در صورت لزوم جابه‌جایی را انجام می‌دهیم. و تا آخر لیست به همین ترتیب جلو می‌رویم. در الگوریتم مرتب‌سازی همواره دو عمل «مقایسه»^۳ و «جابه‌جایی»^۴ انجام می‌شود.

سؤال! وقتی که به انتهای لیست می‌رسیم چه اتفاقی می‌افتد؟

اکنون اعداد مورد نظر را به روش حبابی و به ترتیب صعودی یعنی از کوچک به بزرگ مرتب می‌کنیم:

۱- Bubble Sort Method

۲- Quick Sort Method

۳- Compare

۴- Swap

17	14	8.5	13	20
مقایسه و جابه جایی				
14	17	8.5	13	20
مقایسه و جابه جایی				
14	8.5	17	13	20
مقایسه و جابه جایی				
14	8.5	13	17	20
مقایسه و عدم نیاز به جابه جایی				
14	8.5	13	17	20

فاز اول
مرتب سازی

شکل ۲-۲- مراحل مرتب سازی حبابی، فاز اول

توجه کنید که پس از انجام این تعداد مقایسه و جابه جایی، بزرگ ترین عددی که در لیست وجود دارد خود را به جای اصلی خود، یعنی در انتهای لیست می رساند.

سؤال؟ چرا به این روش مرتب سازی، حبابی گفته می شود؟

هنوز لیست به طور کامل مرتب نشده و باید مجدداً از ابتدای لیست شروع کرده و تا انتهای لیست عمل مقایسه و جا به جایی را انجام دهیم تا عدد بزرگ تر بعدی نیز، خود را به مکان صحیح برساند. در فاز دوم، عدد بزرگ دوم، به مکان یکی مانده به انتهای لیست منتقل می شود.

بنابراین باید، عملیات مقایسه و جابه جایی را بارها تکرار کرد تا در نهایت، لیست به طور کامل

مرتب شود.

سؤال: در مثال ذکر شده چند فاز باید طی شود تا اعداد در جای خود قرار گیرند؟ مراحل بعدی را مانند نمونه بنویسید.

توجه داشته باشید که عمل مرتب سازی یک عمل وقت گیر است به خصوص روش حبابی که یک روش کند است، چون تعداد عمل مقایسه و جا به جایی در آن بسیار زیاد است.

سؤال: در مرتب سازی حبابی، وقت گیرترین و بدترین حالت آرایش اعداد ورودی کدام است؟

اکنون سعی می کنیم الگوریتم را پیاده سازی کنیم. همان طور که در قطعه برنامه زیر مشاهده می کنید در فاز اول ۴ مرحله مقایسه و جابه جایی داریم. چرا؟ برای این که مراحل بالا را پیاده سازی کنیم به حلقه ای نیاز داریم که در هر بار اجرای حلقه، عدد زم را با عدد ۱+زم مقایسه کند و در صورتی که بزرگ تر بود، جابه جایی انجام دهد. دستورات جابه جایی با استفاده از متغیر کمکی سوم را در سال گذشته تمرین کرده اید که یکی از کاربردهای آن در مرتب سازی است. اکنون دستور مربوط به این بخش را می نویسیم:

```
for (int j = 0; j < arr.length - 1; j++)
    if (arr[j] > arr[j + 1])
    {
        temp = arr [j];
        arr[j] = arr[j + 1];
        arr[j + 1] = temp;
    }
```

قسمت شرط پایانی حلقه را فعلاً ننوشته ایم تا به توافق برسیم. اگر در پاسخ به سؤال بالا، مراحل مقایسه و جابه جایی را در فاز دوم و سوم و ... نوشته باشید، متوجه شده اید که حلقه بالایی باید به تعداد فازهای مرتب سازی تکرار شود. بنابراین باید حلقه بالا را درون حلقه دیگری قرار دهیم که مشخص کننده فازهای تکرار است.

سؤال: چرا تعداد تکرار حلقه بیرونی که معرف فازهای مرتب سازی است، یکی کمتر از

تعداد اعداد است؟

```
for (int i = 4; i > 0 ; i--)  
    for (int j = 0; ; j ++)  
        if (arr [j] > arr[j + 1])  
            {  
                temp = arr [j];  
                arr[j] = arr[j + 1];  
                arr[j + 1] = temp;  
            }
```

اکنون نوشتن حلقه‌های مرتب‌سازی حبابی تمام شده است اما هنوز نمی‌دانیم حلقه داخلی که نشان دهنده مراحل در هر فاز است، چه شرطی برای پایان دارد. اگر به واژه حبابی توجه کنیم در می‌یابیم که در پایان هر فاز، یک عنصر در مکان صحیح خود در لیست جای می‌گیرد. پس دیگر نیازی به مقایسه این عنصر نداریم زیرا به درستی در مکان خود قرار گرفته و مرتب شده است. بنابراین در فاز بعدی یک مرحله کمتر از فاز قبل، مقایسه و جابه جایی داریم که با شرط $j < i$ می‌نویسیم. برنامه ۶-۲ را ببینید و با هم کلاس خود برای اعداد قبلی Trace کنید. (برنامه ۶-۲)

سؤال: دستوراتی به برنامه اضافه نمایید تا رتبه‌های اول تا سوم را چاپ نماید.

```

using System;

class BubbleSort
{
    static void Main(string[] args)
    {
        Console.WriteLine("enter the array size: ");
        int arrsize = int.Parse(Console.ReadLine());

        int[] arr;
        arr = new int[arrsize];

        for (int i = 0; i < arrsize; i++)
        {
            Console.WriteLine("enter the {0}th number: ", i+1);
            arr[i] = int.Parse(Console.ReadLine());
        }

        //Bubble Sort.....

        int temp = 0;
        for (int i = arrsize - 1; i > 0; i--)
        {
            for (int j = 0; j < i; j++)
                if (arr[j] > arr[j + 1]) ← مقایسه
                {
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp; } ← و جابجایی
        }

        // show the sorted array
        Console.WriteLine();
        Console.WriteLine(" the sorted array is: ");
        Console.WriteLine("-----");
        for (int i = 0; i < arrsize; i++)
            Console.WriteLine("arr[{0}] = {1}", i+1, arr[i]);
    }
}

```

۷-۲- عمل جستجو در لیست

جستجو پرکاربردترین عملیاتی است که روی لیست‌ها انجام می‌شود و به منظور یافتن عنصری در آرایه صورت می‌گیرد که به آن کلید جستجو گفته می‌شود. در این کتاب ما به توضیح و برنامه نویسی دو نوع جستجو می‌پردازیم.

۱-۷-۲- جستجوی خطی (ترتیبی)^۱:

ساده‌ترین روش جستجو، به صورت ترتیبی می‌باشد که در آرایه‌های کوچک و یا مرتب نشده مورد استفاده قرار می‌گیرد. در این روش جستجو، هر عنصر از آرایه با کلید جستجو مقایسه می‌شود. و برای تعیین اینکه کلید مورد نظر در آرایه وجود دارد یا نه، برنامه باید کلید را با تمام عناصر آرایه مقایسه کند. بنابراین در صورتی که کلید یافت شود نیازی به ادامه جستجو نیست و می‌توان از برنامه خارج شد.

مثال ۷-۲: برنامه‌ای بنویسید که نمره ۲۰ را از میان لیست نمرات کلاس پیدا کرده و اعلام

کند مربوط به نفر چندم لیست است؟

الگوریتم یا روش انجام کار: ابتدا آرایه‌ای از نمرات دانش‌آموزان تعریف و به وسیله حلقه for نمرات را دریافت می‌کنیم. سپس برای جستجوی ترتیبی، با استفاده از حلقه for تک تک عناصر را با کلید جستجو مقایسه کرده و در صورت یافتن کلید به کار حلقه پایان می‌دهیم. برای یافتن کلید از متغیر بولی found کمک می‌گیریم. این متغیر مقدار اولیه false دارد تا زمانی که کلید پیدا شود، در صورتی که کلید یافت نشود، هم چنان مقدار false خواهد داشت.^۲

۱- Liner (Sequential) Search

۲- به چنین متغیرهایی که وضعیت دوحالتی خاموش یا روشن را نشان می‌دهند Flag گفته می‌شود. استفاده از تکنیک Flag در

برنامه‌های زیادی کاربرد دارد.

```

using System;
class Search
{
    static void Main(string[] args)
    {
        Console.WriteLine("enter the array size: ");
        int arrsize = int.Parse(Console.ReadLine());
        Console.WriteLine("-----");
        int[] arr;
        arr = new int[arrsize];
        //.....Initializing array.....
        for (int i = 0; i < arrsize; i++)
        {
            Console.WriteLine("enter the {0}th number: ", i+1);
            arr[i] = int.Parse(Console.ReadLine());
        }
        //.....Linear Search.....
        bool found = false;
        int pos=-1;
        Console.WriteLine("-----");
        Console.WriteLine("enter the search key: ");
        int key = int.Parse(Console.ReadLine());
        Console.WriteLine("=====");
        for (int i = 0; i < arrsize; i++)
            if (arr[i] == key)
            {
                found = true;
                pos = i+1;
                break;
            }
        if (found)
            Console.WriteLine("the key found in position number {0} ", pos);
        else
            Console.WriteLine("the key not found");
        Console.ReadKey();
    }
}

```


۲-۷-۲- جستجوی دودویی (باینری):

روش دیگر جستجوی کلید در بین عناصر آرایه، روش جستجوی دودویی است. فرض کنید می‌خواهید شماره پلاک یک خانه را در یک کوچه بیابید، آنچه باعث می‌شود سریع‌تر خانه مورد نظر را پیدا کنید، مرتب بودن شماره پلاک‌ها در کوچه مورد نظر است.

سؤال: اگر در یک کوچه طولانی دنبال یک خانه بودید و این شماره‌ها مرتب نبود، چه

می‌کردید؟

روش جستجوی دودویی، روی آرایه‌های مرتب قابل اجراست. برای استفاده از روش جستجوی دودویی، در صورتی که آرایه مورد نظر نامرتب باشد، ابتدا باید آن را مرتب کنید. در مثال‌های قبل با روش مرتب‌سازی حبابی آشنا شدید. یکی از کاربردهای مرتب‌سازی در جستجوی دودویی است. این روش جستجو برای آرایه‌های بزرگ مناسب است. چرا؟

سؤال: به طور کلی چرا از این نوع جستجو استفاده می‌کنیم؟

مثال ۲-۸: برنامه‌ای بنویسید که با دریافت کلید جستجو به روش دودویی آن را جستجو

کنید.

الگوریتم یا روش انجام کار: اگر بازی حدس عدد را در کتاب برنامه‌سازی ۱ به خاطر بیاورید، برای حدس عددی که بازیکن اول در ذهن سپرده بود، از روش حرکت کردن در لیست اعداد استفاده می‌کردیم. در ذهن فرد حدس زنده این بود که اعداد مرتب هستند. بنابراین به محض حدس اولین عدد منتظر نتیجه از بازیکن اول می‌ماند تا بگوید عدد حدس زده شده از کلید جستجو بزرگ‌تر است یا کوچک‌تر.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8.5	9	10	11	12.75	14	14.5	15	16	16.5	17	17.5	18	19.5	19.75	20

↑
مقایسه داده مورد نظر با عنصر وسط

شکل ۳-۲- لیست اعداد مرتب به همراه شماره عنصر

به این ترتیب در بازی حدس عدد از تکنیک مشابه جستجوی دودویی استفاده می‌شود. در این

جستجو برای یافتن کلید، ابتدا عنصر وسط لیست با کلید مقایسه می‌شود. این مقایسه ۳ نتیجه در بردارد:

۱- کلید با عنصر وسط لیست یکسان است. بنابراین کلید پیدا شده و در این صورت شماره یا اندیس عنصر وسط آرایه را یادداشت می‌کنیم.

۲- عنصر وسط لیست، کوچک‌تر از کلید است، بنابراین نتیجه می‌گیریم که داده در نیمه سمت چپ نمی‌باشد. در نتیجه باید منطقه مورد جستجو را به نیمه سمت راست آرایه محدود کنیم.

۳- عنصر وسط لیست، بزرگ‌تر از کلید است، بنابراین نتیجه می‌گیریم که داده در نیمه سمت راست لیست نمی‌باشد. در نتیجه باید منطقه مورد جستجو را به نیمه سمت چپ آرایه محدود کنیم.

پس از بررسی مشخص کردن منطقه جدید، مجدداً عمل جستجو را تکرار می‌کنیم یعنی عنصر وسط لیست جدید را مورد بررسی قرار می‌دهیم و یکی از ۳ حالت بالا رخ خواهد داد که بر آن اساس یا داده یافت می‌شود و یا مجدداً عمل جستجو در لیست جدیدی که نصف اندازه قبلی است، تکرار می‌شود.

در روش جستجوی دودویی با یک عمل مقایسه، منطقه مورد جستجو به اندازه نصف، کوچک می‌شود. بنابراین با چند عمل مقایسه یک لیست بزرگ از داده‌ها به منطقه کوچکی ختم می‌شود بنابراین سرعت عملیات جستجو در این روش بسیار بالا است.

برای مثال اگر لیستی دارای ۱۰ هزار عنصر باشد در اولین مرحله ۵۰۰۰ عنصر کنار می‌روند.

در مرحله دوم ۲۵۰۰ عنصر و در مرحله سوم ۱۲۵۰ عنصر کنار می‌روند. بنابراین خواهیم داشت:

مرحله جستجو	تعداد عناصر لیست	مرحله جستجو	تعداد عناصر لیست
۱	۱۰۰۰۰	۲	۵۰۰۰
۳	۲۵۰۰	۴	۱۲۵۰
۵	۶۲۵	۶	۳۱۳
۷	۱۵۷	۸	۷۹
۹	۴۰	۱۰	۲۰
۱۱	۱۰	۱۲	۵
۱۳	۳		

در این جستجو، با حداکثر ۱۳ بار مقایسه، کلید را پیدا می‌کنیم و یا مطمئن می‌شویم که در لیست، چنین داده‌ای وجود ندارد. جستجوی دودویی در مقایسه با روش جستجوی ترتیبی دارای سرعت فوق العاده بالایی است. چرا؟

سؤال: کلید جستجو، در لیست ۱۰۰۰۰ عنصری مثال قبل در روش ترتیبی در چند مرحله جستجو می‌شود؟

نکته

توجه داشته باشید از روش جستجوی دودویی تنها زمانی می‌توانید استفاده کنید که عناصر لیست مرتب باشند. بنابراین اگر لیست مرتب نباشد و بخواهید در لیست بزرگی جستجو کنید، بهتر است ابتدا لیست را مرتب کنید و سپس به روش دودویی جستجو نمایید.

برای اینکه بتوانیم هر بار لیست را نصف کنیم، اندیس‌های آرایه را به سمت منطقه مورد نظر هدایت می‌کنیم. برای این منظور باید وسط لیست را پیدا کنیم و عملیات جابه‌جایی اندیس داشته باشیم.

محاسبه اندیس وسط لیست $middle = (first + last) / 2$;

عملیات جابه‌جایی اندیس زمانی که منطقه جستجو سمت چپ باشد $last = middle - 1$;

عملیات جابه‌جایی اندیس، زمانی که منطقه جستجو سمت راست باشد $first = middle + 1$;

برنامه جستجو را مشاهده کنید. این برنامه برای داده‌های مرتب شده کفایت می‌کند، اما همیشه

برای احتیاط بهتر است قبل از جستجوی دودویی داده‌های وارد شده را مرتب کنید. چرا؟

```

using System;
class BinSearch
{
    static void Main(string[] args)
    {
        Console.WriteLine("enter the array size: ");
        int arrsize = int.Parse(Console.ReadLine());
        Console.WriteLine("-----");
        int[] arr;
        arr = new int[arrsize];
        //.....Initializing arr ay.....
        for (int i = 0; i < arrsize; i++)
        {
            Console.WriteLine("enter the {0}th number: ", i+1);
            arr[i] = int.Parse(Console.ReadLine());
        }
        //.....Binary Search.....
        int last = arrsize - 1;
        int first = 0;
        int middle = (first + last) / 2;
        bool found = false;
        Console.WriteLine("-----");
        Console.WriteLine("enter the search key: ");
        int searchkey = int.Parse(Console.ReadLine());
        Console.WriteLine("=====");
        while (last >= first)
        {
            middle = (first + last) / 2;
            if (arr[middle] == searchkey)
            {
                found = true;
                break;
            }
            else if (arr[middle] > searchkey)
                last = middle - 1;
            else
                first = middle + 1;
        }
        if (found)
            Console.WriteLine("the key found in position number= {0} ", middle);
        else
            Console.WriteLine("the search key not found");
        Console.ReadKey();
    }
}

```

برنامه ۸-۲ در صورتی قابل استفاده است که داده‌ها، مرتب شده وارد شوند. در ورود داده‌ها دقت کنید.

الف) درستی یا نادرستی عبارات زیر را تعیین کنید.

- ۱- جستجوی دودویی فقط بر روی لیست‌های مرتب امکان‌پذیر است.
- ۲- جستجوی خطی برای آرایه‌های بزرگ مناسب است.
- ۳- دستور foreach فقط بر روی آرایه‌ها قابل استفاده است.
- ۴- اندیس اولین عنصر آرایه، ۱ است.
- ۵- از آرایه زمانی استفاده می‌کنیم که مجبوریم چندین بار از داده‌های ورودی در برنامه استفاده کنیم.

ب) جاهای خالی را با عبارات مناسب پر کنید.

- ۶- در عمل جستجو در آرایه به مقداری که جستجو می‌شود..... می‌گویند.
- ۷- در هر الگوریتم مرتب‌سازی همیشه دو عمل..... و انجام می‌شود.

۸- تعداد عناصر آرایه را آرایه می‌نامند.

۹- به هر مکان آرایه، یک آرایه می‌گویند.

ج) به سوالات زیر پاسخ دهید.

- ۱۰- آرایه‌ای برای نام روزهای هفته تعریف کنید.
- ۱۱- آرایه‌ای برای نام ماه‌های سال تعریف کنید.
- ۱۲- برنامه زیر را Trace کنید و در یک خط بنویسید چه عملی انجام می‌دهد.

```
class Program
```

```
{
```

```
static void Main(string[] args)
```

```
{
```

```
int[] b = new int[5];
```

```
int[] a = new int[5];
```

```
int[] c = new int[5];
```

```
for (int i = 0; i <= 4; i++)
```

```

{
    a[i] = int.Parse(Console.ReadLine());
    b[i] = int.Parse(Console.ReadLine());
    if (a[i] > b[i])
        c[i] = a[i];
else
    c[i] = b[i];
    Console.WriteLine(c[i]);
}
Console.ReadKey();
{

```

۱۳- آرایه زیر را در نظر بگیرید و معادل دستورات زیر را با روش دیگری از تعریف آرایه بازنویسی کنید.

```

string[] week = new string[7];
week[0] = "Saturday";
week[1] = "Sunday";
week[2] = "Monday";
week[3] = "Tuesday";
week[4] = "Wednesday";
week[5] = "Thursday";
week[6] = "Friday";

```

۱۴- دستوری بنویسید که تعداد عناصر آرایه week را نمایش دهد.

۱۵- برنامه زیر را Trace کنید و خروجی آن را بنویسید.

```

namespace arrayTest
{
    class Program
    {

```

```

static void Main(string[] args)
{

    int[] arr = { 10, 13, 15, 6, 8, 20 };
    arr[1] = 3;
    for (int i = 0; i < 6; i++)
    {
        arr[i] += 2;
        Console.WriteLine(arr[i]);
    }
    Console.ReadKey();

}
}
}

```

۱۶- با توجه به اعلان آرایه arr به سؤالات زیر پاسخ دهید.

```
int[] arr = { 10, 13, 15, 6, 8, 20 };
```

دستوری بنویسید که مقدار سومین عنصر آرایه را صفر کند.

آرایه arr چقدر حافظه اشغال می‌کند؟

نتیجه اجرای قطعه کد زیر چیست؟

```
int temp = arr[0];
```

```
arr[0] = arr[5];
```

```
arr[5] = temp;
```

۱۷- کدام یک از روش‌های جستجو سریع‌تر است؟ چرا؟

۱۸- در یک آرایه مرتب با ۱۰ عنصر حداکثر با چند مقایسه نتیجه جستجو مشخص می‌شود؟

- ۱- با استفاده از حلقه `foreach` نام روزهای هفته را چاپ کنید.
- ۲- برنامه‌ای بنویسید که نمرات درس برنامه‌سازی یک کلاس ۱۵ نفره را دریافت کند و بیشترین و کمترین نمره کلاس را پیدا کرده و نمایش دهد.
- ۳- برنامه قبل را توسعه دهید تا علاوه بر بیشترین و کمترین نمره کلاس، اعلام کند چندمین دانش‌آموز بیشترین یا کمترین نمره را دارد.
- ۴- شرکت مخابرات می‌خواهد به مشتریانی که در طول یک سال، حداقل ۶ ماه، قبض تلفن همراه خود را قبل از بیستم ماه پرداخت می‌کنند به عنوان جایزه یک بسته مکالمه رایگان ۶۰۰ دقیقه‌ای هدیه دهد. برنامه‌ای بنویسید که برای یک مشتری روز پرداخت قبض در هر ماه را دریافت و اعلام کند که بسته مکالمه رایگان به مشتری تعلق می‌گیرد یا نه؟
- ۵- صاحب یک کارگاه آهنگری می‌خواهد به کارگرانی که بیش از ۳ فرزند دارند، مبلغ ۳۰۰۰۰۰ تومان مساعده پرداخت کند. برنامه‌ای بنویسید که تعداد فرزندان ۱۸ کارگر این کارگاه را دریافت کرده و تعداد کارگرانی که بیش از ۳ فرزند دارند را محاسبه کند و نمایش دهد. سپس کل مبلغ مساعده را نمایش دهد.
- ۶- برنامه قبل را توسعه دهید تا برای هر کارگاه با هر تعداد کارگر و هر مبلغ مساعده قابل استفاده باشد.
- ۷- برنامه سؤال ۵ را با `foreach` بازنویسی کنید.
- ۸- برنامه ۲ را توسعه دهید تا نام و نمره درس برنامه‌سازی یک کلاس ۱۵ نفره را دریافت کند. سپس نام دانش‌آموزانی که کمترین و بیشترین نمره را دارند به همراه نمره آنها نمایش دهد.
- ۹- یک شرکت خصوصی ۳ درصد حقوق کارمندان خود را به عنوان مالیات از آنها کسر می‌کند. برنامه‌ای بنویسید که میزان حقوق ۳۵ کارمند این شرکت را دریافت کند. سپس میزان خالص دریافتی هر کارمند را بعد از کسر مالیات محاسبه و مجدداً در آرایه ذخیره کند.

متن زیر از MSDN با موضوع آرایه برداشت شده است. آن را با کمک هم کلاسی خود ترجمه کنید و به کلاس ارائه نمایید.

Arrays in General

C# arrays are zero indexed; that is, the array indexes start at zero. Arrays in C# work similarly to how arrays work in most other popular languages. There are, however, a few differences that you should be aware of.

When declaring an array, the square brackets ([]) must come after the type, not the identifier. Placing the brackets after the identifier is not legal syntax in C#.

```
int[] table; // not int table[];
```

Declaring Arrays

C# supports single-dimensional arrays, multidimensional arrays (rectangular arrays). The following example shows how to declare single-dimensional array:

Single-dimensional arrays:

```
int[] numbers;
```

Declaring them (as shown above) does not actually create the arrays. In C#, arrays are objects and must be instantiated. The following examples show how to create arrays:

Single – dimensional arrays:

```
int[] numbers = new int[5];
```

واژگان و اصطلاحات انگلیسی فصل دوم

ردیف	واژه انگلیسی	معنی به فارسی
۱	Array	
۲	Binary	
۳	Bubble Sort Method	
۴	Compare	
۵	Element	
۶	Index	
۷	Length	
۸	Liner Search	
۹	Sequeutial Search	
۱۰	Property	
۱۱	Quick Sort Method	
۱۲	Size	
۱۳	Swap	

داده شماری، کلاس و متد

در این فصل علاوه بر آشنایی با نوع داده شماری، با مفاهیم متد و کلاس که از مفاهیم پایه برنامه‌سازی شی‌گرا هستند به صورت مقدماتی آشنا خواهید شد. هدف از این فصل، نوشتن متد یا ایجاد یک کلاس نیست. شناخت اجزای یک کلاس و متد کمک می‌کند در کاربرد آنها بیشتر به تعاریف شان دقت کنید و آنها را درست به کار بگیرید. بدیهی است ایجاد کلاس در کتاب برنامه‌سازی ۳ مورد توجه خاص قرار می‌گیرد.

پس از پایان این فصل انتظار می‌رود که فراگیر بتواند :

- ۱- نوع داده شماری را توضیح دهد.
- ۲- از داده شماری در برنامه‌های خود استفاده کند.
- ۳- متد را تعریف کند و کاربرد آن را بیان نماید.
- ۴- از برخی متدهای آماده در برنامه‌های خود استفاده کند.
- ۵- کلاس، شی و کاربرد آنها را توضیح دهد.
- ۶- شکل کلی یک کلاس و اجزای یک کلاس را نام برده و هر یک را توضیح دهد.
- ۷- کلاس‌های آماده سی‌شارپ را در برنامه‌های خود به کار بندد.

۳-۱- نوع داده شمارشی^۱

تاکنون با داده‌های ساده کار کرده‌اید. داده شمارشی از داده‌های مرکب است که خود شامل یک سری داده است. در شکل ۳-۱ قسمتی از دو برنامه نشان داده شده است که در هر یک از آنها دستور switch استفاده شده است. اگر فرض کنیم هر کدام از برنامه‌ها صحیح کار می‌کنند و کارکرد یکسانی دارند شما کدام یک را ترجیح می‌دهید؟

<pre>switch (player) { case playerRank. Leader: : case PlayerRank. Co-Leade: : case PlayerRank. Elder: : case PlayerRank. Member: : }</pre> <p>قطعه برنامه ب</p>	<pre>switch (player) { case 1: : case 2: : case 1: : case 1: : }</pre> <p>قطعه برنامه الف</p>
--	---

شکل ۳-۱- ساختار switch

همان طور که در شکل ۳-۱ مشاهده می‌کنید با نگاه به قطعه برنامه (الف) متوجه می‌شویم که مقدار متغیر player با اعداد ۱، ۲، ۳ و ۴ مقایسه می‌شود. اما این سؤال پیش می‌آید که این اعداد، مربوط به چه موضوعی هستند؟ آیا شماره بازیکن است؟ آیا فرصت بازی بازیکن است؟ ... و

اما وقتی به قطعه برنامه (ب) نگاه می‌کنیم برداشت دقیق‌تری نسبت به برنامه داریم. مقدار متغیر

^۱ Enumerated Type

player با مقادیر Member ، Elder ، Co_Leader ، Leader که چهار سطح و مقام در بازی^۱ است، مقایسه می‌شود.

خوانایی برنامه (ب) بالاتر از برنامه (الف) است، به دلیل اینکه به جای اعداد ثابت و بی معنی، از کلمات و نام‌های با معنی استفاده شده است. در فصل اول کتاب با مفهوم و کاربرد ثابت‌ها در برنامه آشنا شدید و مشاهده کردید که چگونه خوانایی برنامه را بالا می‌برند. در این بخش نیز با نوع داده شمارشی آشنا می‌شوید که سبب بالا بردن خوانایی برنامه می‌شوند.

نوع داده شمارشی مجموعه‌ای از چند نام دلخواه می‌باشد که حالت‌ها و مقادیر مختلف یک موضوع را نشان می‌دهد. مثلاً برای روزهای هفته به جای اعداد ۱ تا ۷ (یا ۰ تا ۶) از نام‌ها و کلمات معنی دار استفاده می‌کنیم. علاوه بر روزهای هفته، برای نام ماه‌های سال، مقام یا درجه یک بازیکن و مدرک تحصیلی اشخاص نیز می‌توان از نوع داده شمارشی استفاده کرد.

برای تعریف یک نوع داده شمارشی از کلمه کلیدی enum به صورت زیر استفاده می‌شود. نوع دسترسی معمولاً public است و روش نوشتن نام نوع داده نیز مطابق با روش پاسکال است. محل قرارگیری تعریف نوع داده شمارشی، معمولاً خارج از کلاس و در ابتدای برنامه است.

نام دلخواه enum نوع دسترسی

{

لیستی از نام‌ها و کلمات

}

نکته

در لیست نام‌ها و کلمات در نوع داده شمارشی، هر نام با علامت کاما از نام دیگر جدا می‌شود. نقطه ویرگول در این تعریف استفاده نمی‌شود.

در تعریف صفحه بعد نوع داده شمارشی برای نام روزهای هفته میلادی با دسترسی وسیع را مشاهده می‌کنید.

بازی CoC-۱

public enum DayOfWeek

```
{  
    Sunday,  
    Monday,  
    Tuesday,  
    Wednesday,  
    Thursday,  
    Friday,  
    Saturday  
}
```

هر یک از اعضای نوع داده شمارشی معادل با یک عدد ثابت است، این اعداد به طور پیش فرض از عدد صفر شروع می‌شوند و به ترتیب، یک واحد اضافه می‌شوند. مثلاً در تعریف قبل نام Sunday معادل با عدد صفر و نام Monday برابر یک و Saturday برابر ۶ است. اگر مایل باشید می‌توانید عدد دیگری را برای نام‌ها اختصاص دهید. در تعریف زیر نوع داده شمارشی برای ماه‌های یک سال میلادی را مشاهده می‌کنید. در این تعریف January معادل با عدد یک و February برابر دو است.

public enum MonthOfYear

```
{  
    January = 1,  
    February,  
    March,  
    April,  
    May,  
    June,  
    July,  
    August,  
    September,  
    October,  
    November,  
    December  
}
```

سؤال: در قطعه برنامه بالا نام December معادل با چه عددی است؟

همان طور که با نوع داده‌های ساده و ابتدایی مانند `int` یا `float` کار می‌کنیم و می‌توانیم اعداد ثابت و یا متغیری از آن نوع را در برنامه مورد استفاده قرار دهیم دقیقاً همان عملیات را می‌توانیم با نوع داده شمارشی البته با کمی محدودیت انجام دهیم. در این قسمت با ذکر مثال با آنها آشنا می‌شویم.

۱-۱-۳ دسترسی به اعضای یک نوع داده شمارشی :

با نوشتن نام نوع داده شمارشی و سپس نام عضو که این دو با علامت نقطه از هم جدا می‌شوند، می‌توانیم به اعضا یا مقادیر یک نوع داده شمارشی دسترسی پیدا کنیم.

نام عضو. نوع داده شمارشی



`DayOfWeek. Saturday`

شکل ۲-۳ دسترسی به یک عضو نوع داده شمارشی

۲-۱-۳ تعریف یک متغیر از نوع داده شمارشی :

مانند متغیرهای ساده، متغیر نوع شمارشی قادر است فقط یکی از مقادیر نوع شمارشی را در خود جای دهد و به صورت شکل ۳-۳ تعریف می‌شود. اصول نام گذاری متغیرها را رعایت کرده و طبق روش کوهان شتری نام متغیر را می‌نویسیم.

نام متغیر نوع داده شمارشی



`DayOfWeek holiday;`

شکل ۳-۳ تعریف یک متغیر نوع داده شمارشی

۳-۱-۳ مقداردهی متغیرهای شمارشی :

مقداردهی متغیرهای نوع شمارشی معمولاً از طریق دستور انتساب به صورت شکل ۴-۳ انجام می‌شود.

مقدار = نام متغیر

`holiday = DayOfWeek. Sunday;`

شکل ۴-۳ مقداردهی به یک متغیر نوع شمارشی

توجه داشته باشید مقداری که در متغیر نوع شمارشی قرار می‌گیرد باید با نوع آن مطابقت داشته باشد. مثلاً ذخیره عدد ۳ در متغیر holiday به صورت مستقیم امکان‌پذیر نیست و حتماً باید از تبدیل نوع استفاده کنید تا عضوی مطابق عدد ۳ در آن ذخیره گردد.

۴-۱-۳- نمایش مقدار یک عضو یا محتوای یک متغیر نوع شمارشی :
مانند متغیرهای ساده، از متد WriteLine() برای نمایش محتوای متغیری از نوع داده شمارشی و یا یک عضو از آن می‌توان استفاده کرد. شکل ۳-۵ را مشاهده کنید.

نام عضو . نوع داده شمارشی Console. writeLine ("My holiday is " + holiday);
↓ ↓
Console. writeLine ("We should go to school on {0}", Day Of Week. Saturday);

شکل ۳-۵- نمایش مقدار یک نوع داده شمارشی

کار در کارگاه ۱

مثال ۳-۱: برنامه‌ای بنویسید که گروه سنی کاربر را با توجه به سن او طبق جدول ۳-۱ مشخص کند.

جدول ۳-۱- گروه سنی

نام	محدوده سنی
بزرگسالان (adult)	بالای ۱۸ سال
نیمه دوم نوجوانی (late teens)	بالای ۱۵ سال
ابتدای دوران نوجوانی (early teens)	بالای ۱۲ سال
کودکی (Children)	بالای ۷ سال
خردسالی (Infancy)	بالای ۳ سال

الگوریتم یا روش انجام کار : در این برنامه باید ابتدا سن کاربر سؤال شود و سپس با استفاده از دستورات if تودرتو، شرط‌های جدول یک به یک بررسی شود. اگر یک شرط برقرار شد گروه سنی نمایش داده شود. اگر کمی فکر کنیم می‌بینیم خواسته این برنامه، موضوع جدیدی نیست زیرا شبیه این کار را برای نمره یک دانش‌آموز قبلاً انجام داده‌اید. اما در این برنامه از نوع داده شمارشی استفاده می‌کنیم تا با مفهوم این نوع داده و روش استفاده از آن در یک برنامه آشنا شویم.

using System;

```
public enum AgesGroup
{
    adult, earlyTeens, lateTeens, children, infancy
}
```

class userAges Group

```
{
    static void Main (string[] args)
    {
        int age = int. Parse (console. Readline());
        if (age >= 18)
            Console. WriteLine ("your age group is {0}", AgesGroup. adult);
        else if (age >= 15)
            Console. WriteLine ("your age group is {0}", AgesGroup. lateTeens);
        else if (age >= 12)
            Console. WriteLine ("your age group is {0}", AgesGroup. earlyTeens);
        else if (age >= 7)
            Console. WriteLine ("your age group is {0}", AgesGroup. children);
        else if (age >= 3)
            Console. WriteLine ("your age group is {0}", AgesGroup. infancy);
    }
}
```

برنامه ۱-۳ - استفاده از نوع داده شمارشی برای تشخیص گروه سنی

۱- در هنگام نوشتن برنامه، از منوی IntelliSense برای نوشتن سریع نوع داده شمارشی مانند شکل ۶-۳ کمک بگیرید. در برنامه ۱-۳، نوع داده AgesGroup با پنج عضو تعریف شده است، هر

یک از اعضا، معادل یک عدد ثابت است. adult معادل صفر و earlyTeens معادل عدد یک و الی آخر...

```
if (age >= 18)
    console.WriteLine("your age group is {0}", AgesGroup);
```

شکل ۳-۶- مقاردهی از نوع داده شمارشی

۲- در هنگام اجرای برنامه ۱-۳، اعداد مختلفی را به عنوان سن کاربر وارد کنید تا خروجی برنامه را مشاهده کنید.

۳- دستوراتی به ابتدای برنامه اضافه کنید تا نام کاربر سؤال شود و در پایان برنامه همراه با گروه سنی وی چاپ گردد.

سؤال: در مثال ۱-۳ به جای سن کاربر سال تولد او را دریافت کرده و گروه سنی او را مشخص کنید.

مثال ۲-۳: برنامه‌ای بنویسید که مقام یک بازیکن را برحسب تعداد سکه‌ای که در اختیار دارد طبق جدول ۲-۳ مشخص کند.

جدول ۲-۳- درجه یک بازیکن

مقام بازیکن	تعداد سکه
معمولی	تا ۵۰۰۰
ارشد	تا ۱۰۰۰۰
معاون	تا ۱۵۰۰۰
رئیس	از ۱۵۰۰۰ به بالا

الگوریتم و روش انجام کار : در این برنامه باید ابتدا تعداد سکه بازیکن سؤال شود و سپس با استفاده از دستورات if تودرتو، شرط‌های جدول یک به یک بررسی شود. اگر یک شرط برقرار شد درجه بازیکن نمایش داده شود.

در برنامه ۲-۳، نوع داده PlayerRank با چهار عضو تعریف شده است، هر یک از اعضا، معادل یک عدد ثابت است. Leader معادل صفر و Co_Leader معادل عدد یک و الی آخر. . .

```
using System;
```

```
public enum PlayerRank  
{  
    Leader, Co-Leader, Elder, Member  
}
```

```
class playerRankDemo
```

```
{
```

```
    static void Main()
```

```
    {
```

```
        PlayerRank player; ← تعریف متغیر از نوع داده شمارشی
```

```
        Console. Write (“Enter Player coins: ”);
```

```
        string input = Console. ReadLine();
```

```
        long coins = long. Parse (input);
```

```
        if (coins <10000)
```

```
            player = PlayerRank. Member;
```

```
        else
```

```
            if (coins <15000)
```

```
                player = PlayerRank. Elder;
```

```
            else
```

```
                if (coins <20000)
```

```
                    player = PlayerRank. Co-Leader;
```

```
                else
```

```
                    player = PlayerRank. Leader;
```

```
        Console. WriteLine (“The player Rank: {0}”, player);
```

```
    }
```

```
}
```

برنامه ۲-۳- استفاده از نوع داده شمارشی برای تشخیص مقام یک بازیکن

۱- در هنگام نوشتن برنامه، از منوی IntelliSense برای نوشتن سریع نوع داده شمارشی مانند شکل ۳-۷ کمک بگیرید.



شکل ۳-۷- استفاده از نوع داده شمارشی

۲- در هنگام اجرای برنامه ۲-۳، اعداد مختلفی را به عنوان تعداد سکه وارد کنید تا خروجی برنامه را مشاهده کنید.

۳- دستوراتی به ابتدای برنامه اضافه کنید تا نام بازیکن سؤال شود و در پایان برنامه همراه با مقام وی چاپ گردد.

۴- دستوراتی به انتهای برنامه ۲-۳ اضافه کرده تا پس از نمایش مقام بازیکن، تعداد الماس^۱ وی را سؤال کند. (قبل از دریافت تعداد الماس، پیام مناسب چاپ کند.)

۵- دستوراتی به انتهای برنامه اضافه کنید که اگر تعداد الماس وی بیش از ۵۰۰۰ بود به بازیکن پیشنهاد داده شود که آیا مایل هستید با خرج کردن (از دست دادن) تعداد ۵۰۰۰ الماس وضعیت و مقام خود را یک پله افزایش دهید؟ اگر کاربر مایل بود و پاسخ Yes داد در این صورت مقام وی یک پله بالا رود و در غیر این صورت در همان وضعیت باقی بماند. در انتهای برنامه مقام جدید، تعداد سکه و الماس باقیمانده وی با رنگ‌های مناسب نشان داده شود.

۳-۲- استفاده از نوع داده‌های شمارشی آماده در کتابخانه NET.

انواع مختلفی از نوع داده‌های شمارشی آماده در کتابخانه NET وجود دارد، از جمله نوع شمارشی ConsoleColor که بارها برای تعیین رنگ زمینه و قلم صفحه کنسول از آن استفاده کردید. در واقع به جای حفظ کردن تعدادی عدد، از نام رنگ‌ها که قبلاً در ذهن ما نقش بسته است، استفاده می‌کنیم. مثلاً برای اینکه پیامی به رنگ زرد بر روی زمینه آبی نوشته شود، از دستورات زیر

استفاده می کنید :

```
Console. Background Color = ConsoleColor. DarkBlue;  
Console. Foreground Color = ConsoleColor. Yellow;  
Console. WriteLine ("Using colors in console mode.");
```

۳-۳- شیء^۱

در زبان محاوره‌ای، با شنیدن کلمه شیء، به یاد یک جسم بی‌جان می‌افتیم. به عنوان مثال، در محیط مدرسه، روزانه با اشیایی نظیر کتاب، دفتر و توپ فوتبال، سروکار داریم. تصور از کلمه شیء می‌تواند کمی توسعه یابد به طوری که موضوعات دیگر نظیر کارنامه دانش‌آموز و یا فاکتور خرید یک کالا را نیز به عنوان یک شیء در نظر بگیریم.



هر شیء دارای ویژگی‌هایی است که آن را از اشیای دیگر متمایز می‌سازد. مثلاً یک توپ فوتبال دارای ویژگی‌هایی مانند رنگ، طرح و اندازه است که آن را از توپ فوتبال دیگر متمایز می‌سازد و یا در وضعیت ساکن و یا در حال حرکت است.

بر روی هر شیء نیز عملیاتی می‌توان انجام داد. مثلاً در مورد توپ فوتبال می‌توان عمل شوت زدن، پرت کردن و یا گرفتن توپ را انجام داد.

دانش‌آموز را نیز می‌توانیم یک موجود در نظر بگیریم که دارای ویژگی‌هایی مانند نام و نام خانوادگی، نام پدر، شماره ملی است که وی را از بقیه دانش‌آموزان متمایز می‌سازد. همچنین دارای رفتارهای مختلف است. وقتی از وی نامش را سؤال می‌کنند پاسخ می‌دهد، وقتی دوستان و آشنایان را می‌بیند سلام می‌کند. در امتحان شرکت می‌کند، به سؤالات پاسخ می‌دهد



و نمره می‌گیرد، خوشحال می‌شود، می‌خندد.

در زندگی روزمره اشیاء و موجودات مختلف با یکدیگر در ارتباط و تعامل هستند. توپ فوتبال توسط دانش‌آموزی شوت می‌شود، توپ به حرکت درآمده و سرعت می‌گیرد و وضعیت آن تغییر می‌کند. دانش‌آموز دیگری برای گرفتن توپ تلاش می‌کند و در اثر باخت بازی

ناراحت شده و حتی گریه می کند.

دانشمندان رشته ریاضی و کامپیوتر، سعی کرده اند که زبان های برنامه نویسی طراحی کنند که در آنها، برنامه نویس بتواند ویژگی ها و رفتارهای اشیاء و موجودات زندگی واقعی را مدل سازی کند تا بتواند برنامه ای برای حل مسائل در ارتباط با آنها بنویسد.

زبان های برنامه نویسی که در آنها امکان تعریف ویژگی ها و رفتارهای اشیاء و موجودات فراهم شده است، همچنین اجازه می دهند اشیایی بر مبنای ویژگی های تعریف شده، ایجاد شوند و با یکدیگر ارتباط و نسبت به هم واکنش داشته باشند، زبان های شیء گرا^۱ نامیده می شود.

زبان های ++C، جاوا و #C از جمله زبان های شیء گرا هستند، برنامه ای که به این زبان ها نوشته می شود و اجرا می گردد، در واقع از تعدادی شیء تشکیل شده است که با یکدیگر در ارتباط و تعامل هستند.

کلمه شیء در این زبان ها، نه تنها اشیای بی جان، بلکه موجودات جان دار نظیر دانش آموز، کارمند را نیز در بر می گیرد (شاید بهتر باشد به جای کلمه شیء، کلمه موضوع را به کار ببریم که می تواند به هر موجود یا اشیایی اشاره کند).

با توجه به مطالب گفته شده، یک شیء شامل تعدادی ویژگی، وضعیت، رفتار و عملیات است که آن را از اشیای دیگر متمایز می سازد.

مثال ۳-۳: کارنامه یک دانش آموز را به عنوان یک شیء در نظر می گیریم، ویژگی های متمایز کننده و عملیات و رفتارهای انجام شده بر روی آن می تواند چنین باشد:

مقادیری نظیر نام و نام خانوادگی، نام دروس و نمرات هر یک از آنها، از جمله ویژگی هایی است که یک کارنامه را از کارنامه دیگری جدا و قابل تشخیص می کند و برای یک دانش آموز و مدرسه اهمیت دارد. وضعیت یک کارنامه می تواند قبولی، ردی یا مشروطی باشد. عملیات متداولی که بر روی یک کارنامه صورت می گیرد، وارد کردن نمرات در کارنامه، رتبه بندی، چاپ کارنامه و استخراج معدل است که توسط دفتردار مدرسه انجام می شود. برای حل یک مسئله به روش شیء گرا، تشخیص اشیاء و ارتباط آنها با یکدیگر بسیار اهمیت دارد.

ویژگی ها و وضعیت یک شیء به وسیله تعدادی متغیر که فیلد نامیده می شوند، مشخص می شود و رفتارهای اشیاء در قالب متدها تعریف می گردند. بنابراین محل و مکان تعریف فیلدها و متدهای یک شیء در داخل یک کلاس است (شکل ۸-۳).

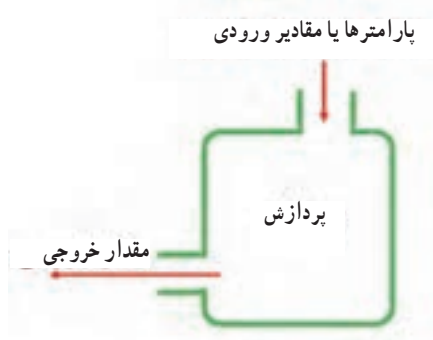


شکل ۸-۳

با توجه به شکل، می‌توان گفت که کلاس تعریف مشخصات، وضعیت و رفتارهای یک شیء را در بردارد و نوع شیء را مشخص می‌کند. بنابراین بهتر است نام کلاس، مطابق با نام شیء باشد که در حال تعریف آن هستیم.

۱-۳-۳- متد :

همان‌طور که می‌دانید متد، مجموعه‌ای از دستورات است که برای انجام یک عمل خاص و حل یک مسئله کوچک به کار می‌رود. به‌طور کلی متد را می‌توان مانند یک دستگاه در نظر گرفت، که از یک طرف موادی وارد آن می‌شود و از طرف دیگر موادی تغییر یافته، از آن خارج می‌شود. این مواد معمولاً داده‌ها هستند که بر روی آنها یک مرحله پردازش صورت می‌گیرد.



شکل ۹-۳- پردازش پارامترها

محل تعریف هر متد در داخل یک کلاس است.

متد جزئی از یک کلاس است. برنامه‌هایی که تاکنون نوشته‌ایم دارای یک کلاس و متد Main بوده‌اند. شکل و ساختار تعریف متد Main() را در شکل ۳-۱۰ مشاهده می‌کنید:

```

عنوان ← static void Main (String[] args)
{
    بدنه متد ←
}

```

شکل ۳-۱۰ عنوان و بدنه متد Main

تعریف متد Main() و یا هر متد دیگر، از دو قسمت تشکیل می‌شود:

۱- عنوان متد

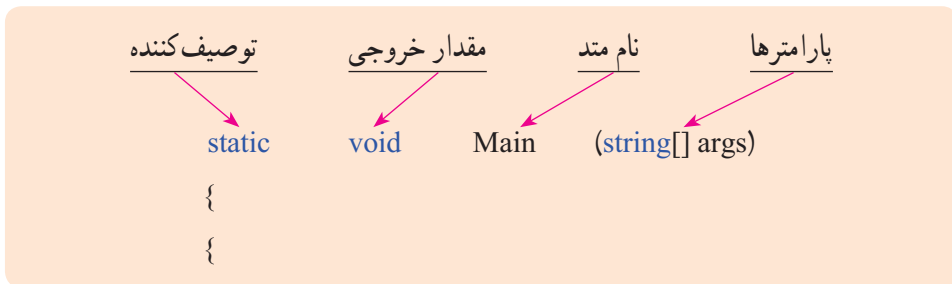
۲- بدنه متد

همان‌طور که با مشاهده شناسنامه یک فرد، اطلاعات مهمی در مورد مشخصات فردی وی بدست می‌آوریم، از روی خط عنوان یک متد نیز، اطلاعات اساسی در مورد متد و طریقه استفاده از آن را متوجه می‌شویم. به خط اول تعریف متد، عنوان متد می‌گویند. در قسمت عنوان متد، مشخصاتی مانند، نام متد، نوع داده خروجی، لیست داده‌های ورودی (پارامترها) و همچنین روش دسترسی^۲ به متد و نحوه استفاده از آن معین و تعریف می‌شود. بنابراین خط عنوان یک متد، بسیار بسیار مهم است و حاوی اطلاعات اساسی در مورد روش استفاده از متد است. در شکل ۳-۱۰ خط عنوان متد Main() که در ویژوال استودیو مشاهده می‌کنید، نشان داده شده است. حال به شرح هر یک از قسمت‌های آن می‌پردازیم.

۱- Heading

۲- Parameters

۳- Access



شکل ۳-۱۱- معرفی خط عنوان متد Main()

بدنه متد همان دستوراتی است که در داخل متد نوشته می‌شود و در برنامه‌های قبلی بارها در متد Main نوشته‌اید و با این دستورات آشنا هستید.

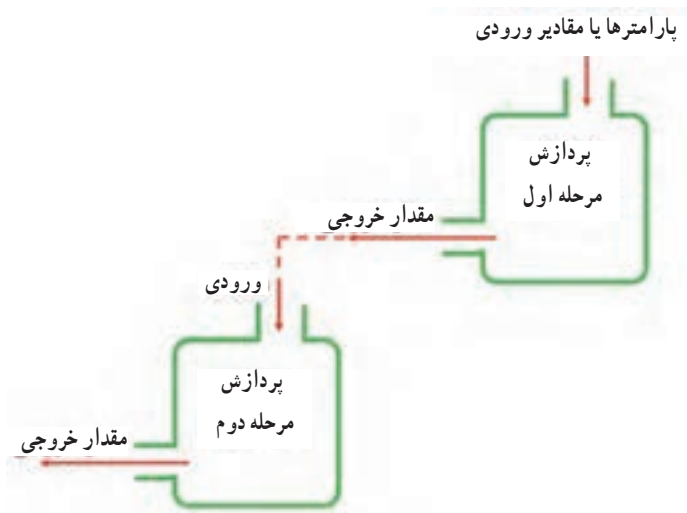
۱-۱-۳- توصیف کننده: خط عنوان یک متد با کلمه‌ای شروع می‌شود که روش ایجاد و محدوده دسترسی به متد را مشخص می‌کند. به عنوان مثال توصیف کننده `static`، روش ایجاد مشخص می‌کند و نشان می‌دهد که به محض اجرای برنامه، چنین متدی ساخته شده و قابل استفاده است. متد `Main()` نیز همواره باید از نوع `static` باشد اگر توصیف کننده `static` را ننویسید، مترجم در ترجمه برنامه خطا می‌دهد. تمام متدهای آماده‌ای که تاکنون استفاده کرده‌اید نظیر `ReadLine()` نیز از نوع استاتیک هستند که در کلاس `Console` از قبل تعریف شده‌اند. استفاده از متدهای `static` ساده است.

توصیف کننده‌های دیگری نظیر `private` و `protected` وجود دارند که به وسیله آنها می‌توان محدوده دسترسی به متد را به یک کلاس محدود کرد و یا با استفاده از توصیف کننده `public` محدوده دسترسی به یک متد را بسیار گسترده و در واقع، بدون محدودیت معین کرد. متدهای `ReadLine()` و `WriteLine()` با توصیف کننده `public` تعریف شده‌اند و به همین دلیل است که می‌توانید در هر برنامه‌ای (هر کلاسی) از آنها استفاده کنید.

نکته

اگر هیچ توصیف کننده‌ای برای تعیین محدوده دسترسی یک متد ذکر نشود، نوع دسترسی `private` به طور پیش فرض برای متد در نظر گرفته می‌شود.

۲-۱-۳- مقدار خروجی یا نوع داده برگشتی^۱: در خط عنوان متد، پس از توصیف‌کننده‌ها، مقدار خروجی یا نوع داده برگشتی متد را باید مشخص کنید. منظور از مقدار خروجی، آنچه که روی صفحه نمایش چاپ می‌شود نیست بلکه مقداری است که از طریق اجرای متد قابل دریافت است و می‌توان آن را در یک متغیر ذخیره کرد و یا به متد دیگری برای ادامه پردازش ارسال کرد. شکل ۱۲-۳ را ملاحظه کنید.



شکل ۱۲-۳- استفاده از خروجی یک متد به عنوان ورودی متد دیگر

به عنوان مثال، می‌خواهیم شعاع دایره را از ورودی گرفته و مساحت آن را محاسبه کنیم. بدین‌منظور باید ابتدا از متد `ReadLine` استفاده کرده و عدد را دریافت کنیم، عدد دریافت شده به صورت `string` است، پس باید با استفاده از متد `Parse()` عدد را به نوع داده اعشاری تبدیل کنیم.

```
string numStr=Console.ReadLine();
float num = float.Parse(numStr);
```

ممکن است متدی مقدار برگشتی یا خروجی نداشته باشد مانند متد `WriteLine()`، در این صورت نوع برگشتی آن `void` یا بدون مقدار خروجی است. معمولاً متد `Main()` نیز، بدون مقدار

برگشتی (void) است^۱.

در مقابل متد WriteLine()، متد ReadLine() را در نظر بگیرید که رشته‌ای را از کاربر دریافت می‌کند و آن را به برنامه تحویل می‌دهد. مقدار برگشتی این متد، یک رشته است و معمولاً آن را در یک متغیر از نوع رشته‌ای ذخیره می‌کنیم.

نکته

ممکن است یک متد، مقدار خروجی و یا مقدار ورودی نداشته باشد.

۳-۱-۳- نام متد: در خط عنوان متد، پس از تعیین نوع برگشتی، باید نام متد را تعیین کنیم. در نام‌گذاری متد باید مانند نام متغیرها، اصول نام‌گذاری شناسه‌ها را رعایت کنیم. معمولاً نام متد به صورت یک فعل امری نام‌گذاری می‌شود. این نام بهتر است بیان‌کننده نوع کاری باشد که متد، قصد انجام آن را دارد. برای مثال متد Clear برای پاک کردن صفحه نمایش و متد Write برای نوشتن روی صفحه نمایش است.

۴-۱-۳- لیست پارامترهای ارسالی: در خط عنوان متد، پس از تعیین نام، باید پارامترها یا داده‌هایی که از طرف برنامه به متد داده می‌شود (اصطلاحاً به متد ارسال می‌شود) را مشخص کنیم. مثلاً برای متد Write پارامتر ورودی می‌تواند یک رشته باشد که باید روی صفحه کنسول نمایش داده شود. بعضی از متدها نیز پارامتر ورودی ندارند مانند متد ReadLine().
به عنوان مثال، خط عنوان متد WriteLine() را بررسی می‌کنیم:

```
public static void WriteLine (string value)
```

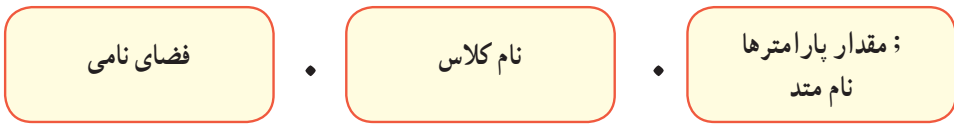
↓ خروجی ↓ نوع داده ورودی

با توجه به خط عنوان، نام متد WriteLine است و ورودی آن یک رشته مانند value است. این متد خروجی ندارد و با دسترسی وسیع و از نوع استاتیک است.

سؤال: با توجه به خط عنوان متد WriteLine()، توصیف‌کننده‌های آن کدام است؟

۱- البته خروجی متد Main () می‌تواند عدد صحیح int نیز باشد.

۱-۳-۳-۲ روش استفاده یا فراخوانی^۱ متد: باید بتوانیم از متد در برنامه استفاده کنیم و با به اصطلاح آن را فراخوانی کنیم. چنانچه متدی از نوع استاتیک باشد فراخوانی آن بسیار ساده است به شرط اینکه خط عنوان آن را بدانیم. برای اطلاع از خط عنوان متدهای آماده Net. می توانید به راهنمای آنها مراجعه کنید. پس از اطلاع از عنوان متد، کافی است فضای نامی و نام کلاس را قبل از نام متد ذکر کنید. این سه قسمت با علامت نقطه از یکدیگر جدا می شوند. اگر متد دارای پارامتر است باید مقدار پارامترها را نیز در داخل پرانتز معین کنید و در انتها علامت نقطه ویرگول را بنویسید.



فرض کنید می خواهیم صدایی با فرکانس ۲۰۰۰ هرتز به مدت ۱ ثانیه ایجاد کنیم. در کتاب برنامه سازی ۱ از متد `Beep()` برای ایجاد صدا استفاده کردید. این متد صدا را با فرکانس معین در مدت زمان مشخص تولید می کند. خط عنوان این متد به صورت زیر است:

`public static void Beep(int frequency, int duration)`

این متد به صورت استاتیک و در کلاس `Console` تعریف شده است و خروجی ندارد. این کلاس در فضای نامی `System` قرار دارد، بنابراین فراخوانی متد `Beep()` چنین است:

`System.Console.Beep(2000,1000)`

سؤال؟ در چه صورت می توانیم دستور بالا، را به صورت زیر خلاصه کنیم؟

`Console.Beep(2000,1000)`

فرض کنید می خواهیم نام کاربر را از ورودی دریافت کنیم. بارها از متد `RedLine` برای دریافت از ورودی استفاده کرده اید. خط عنوان این متد به صورت زیر است:

`public static string ReadLine()`

این متد ورودی ندارد و خروجی آن از نوع `string` است و به صورت استاتیک و در کلاس

^۱ Method call

Console تعریف شده است و در فضای نامی System قرار دارد، بنابراین فراخوانی متد(ReadLine) چنین است :

```
System.Console.ReadLine()
```

سؤال: در چه صورت می توانیم دستور بالا، را به صورت زیر خلاصه کنیم؟

```
Console.ReadLine()
```

با اجرای هر یک از دستورات بالا، نام کاربر از ورودی دریافت می شود ولی برای استفاده در جایی ذخیره نمی شود. برای استفاده از نام کاربر، باید خروجی متد را در یک متغیر ذخیره کنیم و یا به متد WriteLine() برای چاپ بر روی صفحه نمایش ارسال کنیم. برای ذخیره نام کاربر در یک متغیر می توانیم چنین دستوری بنویسیم :

```
string userName = Console.ReadLine();
```

و همچنین برای نشان دادن نام کاربر بر روی صفحه نمایش، چنین دستوری می نویسیم :

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

سؤال: چرا متغیر userName از نوع string تعریف شده است؟

- الف) در سؤالات چند گزینه‌ای زیر پاسخ صحیح را انتخاب نمایید.
- ۱- چه چیز یک شی را از اشیا دیگر متمایز نمی‌کند؟
الف) مشخصه ب) رفتار ج) عملیات د) توصیف کننده
 - ۲- کدام گزینه در مورد متد صحیح نیست؟
الف) محل تعریف متدهای یک شی داخل کلاس است.
ب) نحوه دسترسی به متد در خط عنوان متد تعیین می‌شود.
ج) یک متد می‌تواند پارامتر ورودی نداشته باشد.
د) اگر محدوده دسترسی به متد تعیین نشود به طور پیش فرض public در نظر گرفته می‌شود.
 - ۳- کدام گزینه در مورد نوع داده شمارشی صحیح است؟
الف) محل قرارگیری تعریف نوع داده شمارشی در متد Main است.
ب) شماره معادل با اولین عضو نوع داده شمارشی، عدد یک است.
ج) برای جدا کردن نام‌ها در نوع داده شمارشی، علامت , به کار می‌رود.
د) نوع دسترسی به یک نوع داده شمارشی معمولاً private به کار می‌رود
ب) جاهای خالی را با عبارات مناسب پر کنید.
 - ۴- مجموعه‌ای از چند نام دلخواه که حالت‌ها و مقادیر مختلف یک موضوع را نشان می‌دهند نام دارد.....
 - ۵- اگر یک متد مقداری برنمی‌گرداند، نوع خروجی آن را تعیین می‌کنیم.
 - ۶- کلمه‌ای که خط عنوان متد با آن شروع شده و محدوده دسترسی به متد را تعیین می‌کند نام دارد.
 - ج) به سؤالات زیر پاسخ دهید.
 - ۷- ویژگی‌ها و رفتار و عملیات قابل انجام بر روی هر یک از اشیا زیر را تعیین کنید.
کتاب، تلویزیون، دستگاه بازی
 - ۸- با یک مثال نشان دهید که خروجی یک متد می‌تواند ورودی یک متد دیگر باشد.
 - ۹- روش ایجاد اکثر متدهای آماده کتابخانه NET چیست؟
 - ۱۰- چرا روش ایجاد متد Main() باید static باشد؟

۱۱- اشکال برنامه زیر چیست؟ با یک تغییر ساده در برنامه اشکال را برطرف کنید.

```
using System;
namespace checkError
{
    class Program
    {
        static void Main(string[] args)
        {
            enum Sesean
            { spring, summer, fall, winter }

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

متن زیر از MSDN با موضوع داده‌های شمارشی برداشت شده است.

enum

The enum keyword is used to declare an enumeration, a distinct type that consists of a set of named constants called the enumerator list.

Usually it is best to define an enum directly within a namespace so that all classes in the namespace can access it with equal convenience. However, an enum can also be nested within a class or struct.

By default, the first enumerator has the value 0, and the value of each successive enumerator is increased by 1. For example, in the following enumeration, Sat is 0, Sun is 1, Mon is 2, and so forth.

```
enum Days {Sat, Sun, Mon, Tue, Wed, Thu, Fri};
```

Enumerators can use initializers to override the default values, as shown in the following example.

```
enum Days {Sat=1, Sun, Mon, Tue, Wed, Thu, Fri};
```

In this enumeration, the sequence of elements is forced to start from 1 instead of 0. However, including a constant that has the value of 0 is recommended. For more information, see Enumeration Types (C# Programming Guide).

Every enumeration type has an underlying type, which can be any integral type except char. The default underlying type of enumeration elements is int. To declare an enum of another integral type, such as byte, use a colon after the identifier followed by the type, as shown in the following example.

```
enum Days: byte {Sat=1, Sun, Mon, Tue, Wed, Thu, Fri};
```

The approved types for an enum are byte, sbyte, short, ushort, int, uint, long, or ulong.

واژگان و اصطلاحات انگلیسی فصل سوم

ردیف	واژه انگلیسی	معنی به فارسی
۱	Access	
۲	Behavior	
۳	Enumerated Type	
۴	Gem	
۵	Heading	
۶	Member	
۷	Method Call	
۸	Modifier	
۹	Object Oriented Language	
۱۰	Parameter	
۱۱	Property	
۱۲	Return Type	

کار با متدها و کلاس‌های آماده

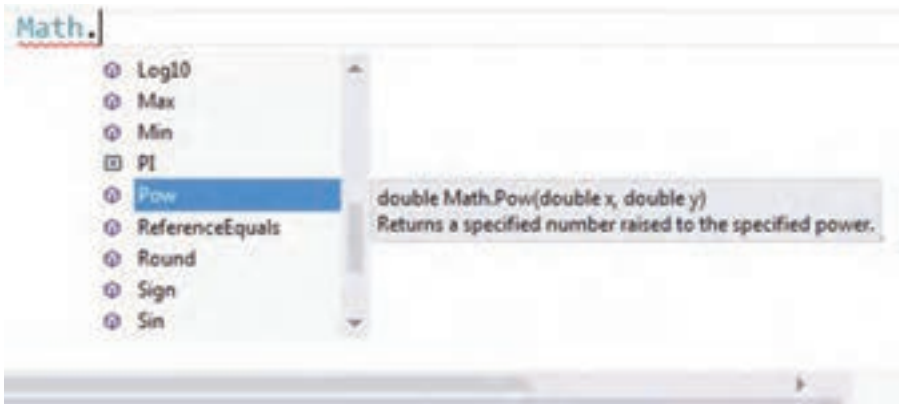
تاکنون با تعدادی از متدها و کلاس‌های سی شارپ کار کرده‌اید. این متدها و کلاس‌ها، از قبل آماده و درون ساختار سی شارپ تعبیه شده است که می‌توانیم از آنها در برنامه‌های خود استفاده کنیم و به این طریق برنامه خود را به راحتی و با قدرت بیشتر بنویسیم. کتابخانه NET Framework. به فراوانی دارای این کلاس‌ها و متدهاست. در این فصل برای نمونه به سه کلاس اشاره می‌کنیم که کاربردهای زیادی در برنامه‌های شما خواهد داشت.

پس از پایان این فصل انتظار می‌رود که فراگیر بتواند:

- ۱- اجزای خط عنوان یک متد را توضیح دهد.
- ۲- ورودی‌ها و خروجی‌های یک متد را نام ببرد.
- ۳- با کلاس math و متدهای پرکاربرد آن برنامه بنویسد.
- ۴- از کلاس array و متدهای sort و search این کلاس در برنامه‌های خود استفاده کند.
- ۵- کاربرد کلاس string را بیان نماید و با استفاده از متدهای این کلاس برنامه بنویسد.

۱-۴- کلاس Math

برای انجام محاسبات ریاضی علاوه بر چهار عمل اصلی، نیاز به عملگرهای دیگری نظیر توان و یا عمل جذر و غیره می‌باشد. زبان C# از طریق کتابخانه .Net، انواع توابع ریاضی، مثلثاتی و لگاریتمی را برای برنامه‌نویس فراهم نموده است. این توابع به صورت متدهای استاتیک در کلاس Math تعریف شده‌اند. در این قسمت با تعدادی از آنها که ساده می‌باشند، آشنا می‌شویم. البته قرار نیست اینجا ریاضیات کار کنیم، اما خواهید دید به راحتی می‌توانید کار محاسبه را به برنامه خود واگذارید. به محض نوشتن نام کلاس Math و علامت نقطه، لیستی از متدهای کلاس Math به وسیله IntelliSense نشان داده می‌شود.



شکل ۱-۴- لیست متدهای کلاس Math

با حرکت در لیست مربوطه، در مورد هر یک از متدها توضیحی ظاهر می‌شود. این توضیح شامل خط عنوان متد و عملکرد آن است. در واقع شناسنامه متد نشان داده می‌شود. در ادامه به توضیح برخی متدهای کلاس Math می‌پردازیم.

۱-۴-۱- متد Pow()

ورودی‌های این متد، دو عدد x و y از نوع اعشاری با دقت بالا است و متد نیز یک عدد اعشاری دقت بالا که حاصل x به توان y است را تولید نموده و به برنامه اصلی باز می‌گرداند. مثلاً برای محاسبه عبارت ۵ به توان ۲ کافی است اعداد ۵ و ۲ را به این متد ارسال کنیم و این متد نیز پس از اجرا عدد ۲۵ را بر می‌گرداند. از مقدار برگشتی می‌توان استفاده کرد. به عنوان مثال آن را داخل یک متغیر بریزیم و در یک عملیات دیگر مانند چاپ از آن بهره بگیریم:

```
double number = Math.Pow(5, 2);
```

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

البته می‌توانیم به صورت مستقیم نیز مقدار برگشتی را چاپ کنیم:

```
Console.WriteLine(Math.Pow(5, 2));
```

و یا مقدار خروجی را استفاده نکرد و دور ریخت!

```
Math.Pow(5, 2);
```

عملکرد متد Pow را می‌توان به صورت شکل زیر نشان داد:



شکل ۲-۴- عملکرد متد Pow

مثال ۴-۱: با اجرای برنامه زیر، عدد ۲ به توان اعداد مختلف (از توان صفر تا ۱۶) روی صفحه نمایش نشان داده می‌شوند.

```
class mathDemo
{
    static void Main ( )
    {
        for (int i = 0; i <= 16; i++)
        {
            double number = Math.Pow(2, i);
            Console.WriteLine(number);
        }
    }
}
```

برنامه ۴-۱- نمایش توان‌های مختلف عدد ۲

کلاس **Math** در فضای نامی **System** تعریف شده است و برای استفاده از متدهای آن باید نام **System** قبل از نام کلاس نیز ذکر گردد و یا در ابتدای برنامه با دستور **using**، حوزه **System** به برنامه معرفی شود.

در جدول زیر لیستی از متدهای ریاضی نشان داده شده است. این متدها از کلاس **Math** هستند.

جدول ۱-۴- برخی متدهای ریاضی از کلاس **Math**

نام متد	کاربرد	شکل کلی	مثال
<u>Cos</u>	کسینوس یک زاویه برحسب رادیان را حساب می‌کند.	Math.Cos (عدد)	Math.Cos(10)
<u>Log</u>	لگاریتم عدد را در مبنای تعیین شده حساب می‌کند.	Math.Log (عدد، مبنا)	Math.Log(16,2)
<u>Max</u>	عدد بزرگ‌تر از بین دو عدد را پیدا می‌کند.	Math.Max (عدد، عدد)	Math.Max(10,20)
<u>Min</u>	عدد کوچک‌تر از بین دو عدد پیدا می‌کند.	Math.Min (عدد، عدد)	Math.Min(10,20)
<u>Pow</u>	حاصل عدد به توان تعیین شده را حساب می‌کند.	Math.Pow (توان، عدد)	Math.Pow(10,3)
<u>Round</u>	عدد را گرد می‌کند.	Math.Round (عدد)	Math.Round(10.7)
<u>Sin</u>	سینوس یک زاویه برحسب رادیان را حساب می‌کند.	Math.Sin (عدد)	Math.Sin(20)
<u>Sqrt</u>	جذر عدد را حساب می‌کند. همان رادیکال خودمان.	Math.Sqrt (عدد)	Math.Sqrt(100)
<u>Tan</u>	تانژانت یک زاویه برحسب رادیان را حساب می‌کند.	Math.Tan (عدد)	Math.Tan(20)
<u>Truncate</u>	بخش اعشار عدد را حذف می‌کند.	Math.Truncate (عدد)	Math.Truncate (10.5)

۲-۱-۴ عدد π : برای محاسبه مساحت دایره نیاز به عدد π است. این عدد گنگ از حاصل تقسیم محیط دایره بر اندازه قطر به دست می‌آید و برای تمامی دایره‌ها ثابت و در حدود $3/14$ است^۱. اگر نیاز به دقت بیشتری در محاسبات داشتید می‌توانید از ثابت π که در کلاس `Math` تعریف شده است استفاده کنید.

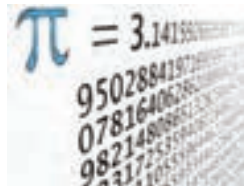
`Math.PI`

توجه داشته باشید که `PI` متد نیست بلکه یک ثابت است. می‌توانید مقدار آن را در یک متغیر اعدادی ذخیره کنید و یا به متد `WriteLine()` برای چاپ ارسال کنید:

`Console.WriteLine(Math.PI);`

فرض کنید: می‌خواهیم اندازه مساحت دایره‌ای را محاسبه و نمایش دهیم که کاربر اندازه شعاع آن را وارد می‌کند.

الگوریتم و روش انجام کار: برای محاسبه مساحت^۲ دایره از فرمول πr^2 استفاده می‌کنیم که منظور از r شعاع^۳ دایره است.



```
// Area of a Circle =  $\pi r^2$ 
```

```
double area = Math.PI * Math.Pow(radius, 2);
```

کنجکوی

بررسی کنید متد `Abs` چه عملی انجام می‌دهد؟

^۱ $\pi = 3/14159265358979323846$

^۲ Area

^۳ Radius

۱- برنامه زیر را در محیط VS تایپ کرده و نتیجه اجرای آن را مشاهده کنید :

```
using System;
class Program
{
    static void Main ( )
    {
        Console.WriteLine (Math.Max(100,200)); //200
        Console.WriteLine (Math.Max(-100,-200) );

        Console.WriteLine (Math.Min(100,200) ); //100
        Console.WriteLine (Math.Min(-100,-200) );

        Console.WriteLine (Math.Pow (2, 3) ); //8
        Console.WriteLine (Math.Pow(-2, 3) );

        Console.WriteLine (Math.Round(18.5) ); //19
        Console.WriteLine (Math.Round(18.25) );

        Console.WriteLine (Math.Sqrt (225) ); //15
        Console.WriteLine (Math.Sqrt (200) );

        Console.WriteLine (Math.Truncate (1.23) ); //1
        Console.WriteLine (Math.Truncate (1.54) );
    }
}
```

برنامه ۲-۴- آزمایش متدهای کلاس Math

۲- خروجی هر خط برنامه را بنویسید. نتیجه خروجی بعضی از خطوط برنامه نوشته شده

است.

مثال ۲-۴: برنامه‌ای بنویسید که ۱۰ عدد را بگیرد و کمترین آنها را نمایش دهد.

الگوریتم و روش انجام کار: برنامه محاسبه کمترین عدد را سال گذشته نوشته‌اید. با توجه به اینکه در فصل ۲ آرایه را یاد گرفته‌اید، این برنامه را با آرایه می‌نویسیم. برای این منظور پس از تعریف متغیر آرایه‌ای، برای دریافت اعداد از حلقه for کمک می‌گیریم. سپس اولین عدد را به عنوان کمترین عدد لیست در متغیر min می‌ریزیم و از عدد دوم تا انتهای لیست، عمل مقایسه و ریختن در متغیر min را در صورت نیاز انجام می‌دهیم. متد Min کلاس math به ما کمک می‌کند تا عمل مقایسه زیر را با دستور انتساب جایگزین کنیم (برنامه ۳-۴).

```
if (min < arr [i])
    min = arr[i];
```

سؤال: با توجه به برنامه زیر توضیح دهید کدام دستور جایگزین شرط بالاست.

```
using System;
namespace MinAndMax
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] arr;
            arr = new int[10];
            for (int i = 0; i <=9; i++)
            {
                Console.WriteLine("enter the {0}th number: ", i+1);
                arr[i] = int.Parse(Console.ReadLine());
            }
            int min = arr[0];
            for (int i = 1; i <= 9; i++)
                min = Math.Min(min, arr[i]);
```



```

        Console.WriteLine("minimum is {0}", min);
        Console.ReadKey();
    }
}
}

```

برنامه ۳-۴ - محاسبه کوچک ترین عدد

سؤال: اگر بخواهیم بیشترین عدد را هم در این برنامه نمایش دهیم چه تغییراتی باید انجام

دهیم؟

۲-۴ - کلاس string

همان طور که در برنامه می توان متغیری برای نگهداری داده های عددی تعریف کرد، برای داده های رشته ای نیز می توان متغیری از نوع string تعریف کرد. این نوع داده در واقع یک کلاس است و متغیری که از این نوع تعریف می شود اشاره به ردیفی از کاراکترها می کند.

مثال ۳-۴: برای نگهداری کلمه Mohammad از یک متغیر رشته ای به صورت زیر

استفاده می کنیم :

```
string name = "Mohammad";
```

7	d
6	a
5	m
4	m
3	a
2	h
1	o
0	M

شکل ۴-۴ - ساختمان یک رشته

برای آنکه به حروف یک رشته دسترسی داشته باشیم می‌توانیم مانند آرایه‌ها، از اندیس حروف استفاده کنیم.

[شماره کاراکتر] نام رشته متنی

توجه داشته باشید اندیس از صفر شروع می‌شود. بنابراین `name[5]` کاراکتر ششم رشته `name` را در خود جای داده است.

نکته

تغییر ناپذیری یک رشته: پس از اینکه یک رشته در حافظه ایجاد شد، محتویات آن قابل تغییر نیست. به عبارت دیگر یک رشته پس از ایجاد، تغییر ناپذیر است. شاید از خود پرسید چرا می‌توانیم به یک رشته مقداری دیگر دهیم؟ پاسخ این است که هر بار که متغیر رشته‌ای مقدار دیگری می‌گیرد مانند این است که دوباره تعریف شده است! اما ما نمی‌توانیم مثلاً یک حرف از آن را تغییر دهیم.

اگر متغیری را از نوع `string` تعریف نماییم، این متغیر دارای متدهایی خواهد بود که می‌توانند عملیات سودمندی را انجام دهند. مثلاً متدی به نام `ToLower()` قادر است حروف انگلیسی موجود در رشته را به حروف کوچک تبدیل کند و در مقابل آن، متد `ToUpper()` می‌تواند حروف انگلیسی رشته را به حروف بزرگ تبدیل کند.

جدول ۲-۴- برخی متدهای رشته‌ای از کلاس String

نوع برگشتی	شرح کار متد	متد
bool	مقایسه دو رشته	CompareTo()
int	موقعیت وجود یک کاراکتر در رشته را برمی‌گرداند.	IndexOf()
string	تبدیل تمام کاراکترهای یک رشته به حروف کوچک	ToLower ()
string	تبدیل تمام کاراکترهای یک رشته به حروف بزرگ	ToUpper ()
string	درج یک کاراکتر یا یک رشته در درون یک رشته دیگر	Insert()
int	تعداد کاراکترهای یک رشته را برمی‌گرداند.	Length
string	یک کاراکتر در یک رشته را با کاراکتر یا یک رشته دیگری عوض می‌کند.	Replace()

نکته

این متدها روی خود رشته تأثیر نمی‌گذارند بلکه یک رشته دیگر برمی‌گردانند. بنابراین محتوای متغیر رشته‌ای را تغییر نمی‌دهند.

کار در کارگاه ۳

مثال ۴-۴: کار با متدهای رشته‌ای :

۱- عملکرد متدهای مختلف کلاس string را با تایپ برنامه صفحه بعد مشاهده کنید :

```

using System;
class Program
{
    static void Main(string[] args)
    {
        string s = " This is a String! ";
        Console.WriteLine ("the main string : \"{0}\"", s);
        Console.WriteLine (" ToLower: \"{0}\"", s.ToLower());
        Console.WriteLine (" ToUpper: \"{0}\"", s.ToUpper());
        Console.WriteLine (" Insert: \"{0}\"", s.Insert(0, "Note,") );
        Console.WriteLine (" Length: {0}", s.Length);
        Console.WriteLine (" index of 'i': {0}", s.IndexOf('i'));
        Console.WriteLine (" Remove: \"{0}\"", s.Replace("This is", "we have"));
        Console.ReadKey();
    }
}

```

برنامه ۴-۴- آزمایش متدهای کلاس String

کار در کارگاه ۴

مثال ۴-۵: برنامه‌ای بنویسید که یک نام کاربری و گذرواژه را بگیرد. اگر نام کاربری user و گذرواژه "pass" وارد شود پیام ورود موفقیت‌آمیز داده شود. همان‌طوری که در پست الکترونیکی و برنامه‌های مختلف دیده‌اید نام کاربری به کوچک یا بزرگی حروف حساس نیست ولی گذرواژه باید دقیقاً مطابقت داشته باشد.

الگوریتم و روش انجام کار: در این برنامه پس از دریافت نام کاربری و گذرواژه، باید بررسی شود که برابر با مقدار خواسته شده می‌باشد یا خیر. برای آنکه بررسی نام کاربری حساس به نوع حروف نباشد نامی که کاربر وارد می‌کند را به رشته‌ای با حروف کوچک تبدیل می‌نماییم. چرا؟

```

using System;
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("enter username: ");
        string username = Console.ReadLine();
        Console.WriteLine("enter password: ");
        string password = Console.ReadLine();
        if (username.ToLower() == "user" && password == "pass")
            Console.WriteLine("your login was successful. Welcome!");
        else
            Console.WriteLine("username or password is wrong");
        Console.ReadKey();
    }
}

```

برنامه ۴-۵- بررسی نام کاربری حساس به کوچکی و بزرگی حروف

سؤال؟ اگر بخواهیم از ToUpper به جای ToLower استفاده کنیم کد چه تغییری

می‌کند؟

سؤال؟ این برنامه را طوری تغییر دهید که تا وارد شدن نام و گذرواژه صحیح، با دادن

پیام مناسب، عمل دریافت تکرار شود.

۴-۳- کلاس Array

در فصل ۲ با مبحث آرایه و کاربردهای آن آشنا شدید. برخی از برنامه‌هایی که با استفاده از آرایه‌ها نوشتیم دارای کدهای طولانی و وقت‌گیری است که با کلاس Array و متدهای آن قابل جایگزینی است. از ویژگی‌های این کلاس، وجود متدهای مختلف استاتیک، برای عملیات بر روی آرایه‌ها است. این متدها کاملاً آزمایش شده و مطمئن هستند. برای مثال عمل جستجوی خطی، دودویی و مرتب‌سازی آرایه با استفاده از متدهای کلاس Array به راحتی قابل انجام است. در جدول ۴-۳ لیست این متدها و توضیحات آنها را می‌بینید.

جدول ۳-۴- برخی متدهای کلاس Array

نام متد	کاربرد	شکل کلی	مثال
Sort	مرتب سازی لیست	Array.Sort (نام آرایه)	Array.Sort (names)
Reverse	وارونه کردن عناصر آرایه	Array.Reverse (نام آرایه)	Array.Sort (names)
IndexOf	جستجوی یک مقدار در آرایه و برگرداندن مکان اولین مورد پیدا شده	Array.IndexOf (نام آرایه، مقدار)	Array.IndexOf(names,"ali")
LastIndexOf	جستجوی یک مقدار در آرایه و برگرداندن مکان آخرین مورد پیدا شده	Array.LastIndexOf (نام آرایه، مقدار)	Array.LastIndexOf(names,"ali")
BinarySearch	جستجوی یک مقدار در آرایه مرتب شده صعودی و برگرداندن مکان اولین مورد پیدا شده	Array.BinarySearch (نام آرایه، مقدار)	Array.BinarySearch(names, name);
Copy	کپی کردن عناصر یک آرایه در آرایه دیگر از اولین عنصر تا تعداد تعیین شده	Array.Copy (تعداد، نام آرایه مبدأ عناصر، نام آرایه مقصد)	Array.BinarySearch(names, family,10);
Clear	پاک کردن و تنظیم مقدار عناصر تعیین شده آرایه یا مقدار پیش فرض نوع داده آرایه	Array.Clear (تعداد عناصر، شروع اندیس، نام آرایه)	Array.BinarySearch(names, 4,10);

کار در کارگاه ۵

مثال ۶-۴: برنامه‌ای بنویسید که اسامی دانش‌آموزان یک کلاس را دریافت کرده و سپس

آنها را براساس نام، مرتب کرده و نمایش دهد.

الگوریتم یا روش انجام کار: ابتدا تعداد دانش‌آموزان از کاربر دریافت و به همان میزان،

آرایه‌ای ایجاد و اسامی از کاربر دریافت می‌شود. در مرحله بعد، اسامی وارد شده مرتب و در نهایت

اسامی چاپ می‌شود.

```
using System;
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("enter number of students: ");
        byte count = byte.Parse(Console.ReadLine());
        string[] studentName = new string[count];
        for (int i = 0; i < count; i++)
        {
            Console.WriteLine("Student Name {0}: ", i + 1);
            studentName[i] = Console.ReadLine();
        }
        Array.Sort(studentName);
        Console.WriteLine("students in name order: ");
        for (int i = 0; i < count; i++)
            Console.WriteLine("{0,3} ==> {1,-25}", i + 1, studentName[i]);
        Console.ReadKey();
    }
}
```

برنامه ۴-۶- مرتب‌سازی اسامی با استفاده از متد Sort

۲- آرایه را پس از مرتب شدن برعکس نمایش دهید.

۳- در انتهای برنامه و قبل از دستور Console.ReadKey() قطعه برنامه زیر را که نتیجه

جستجوی دودویی و خطی یک نام در آرایه است را اضافه کنید.

```
Console.WriteLine("\n enter a name for search in students: ");
string name = Console.ReadLine ();
Console.WriteLine("Binary Search Result: " + Array.Binary Search
(studentName, name));
Console.WriteLine("linear search Result: " + Array.Index Of(student Name,
name));
```

۴- در اجرای برنامه، یکی از اسامی موجود در لیست دانش آموزان را برای جستجو وارد کنید. نتایج را مشاهده کنید. نتیجه جستجوی دودویی ۱- خواهد بود. این پاسخ به معنای یافت نشدن نام مورد نظر است. چرا نتیجه چنین است؟

- ۱- هر متد از سمت راست را با توضیح سمت چپ مطابقت دهید (یک توضیح اضافی است).
- | | |
|----------------------|------------------------|
| ۱- گرد کردن عدد | Math.Sqrt (الف) |
| ۲- مرتب کردن آرایه | Math.Pow (ب) |
| ۳- وارون نمودن آرایه | Math.Round (ج) |
| ۴- توان رساندن عدد | Array.BinarySearch (د) |
| ۵- جستجوی خطی کلید | Array.Sort (و) |
| ۶- محاسبه جذر عدد | Array.Reverse (ه) |
- ۷- جستجوی دودویی در یک آرایه مرتب

۲- با متدهایی که تاکنون شناخته‌اید جدول زیر را تکمیل کنید.

کلاس array	کلاس string	کلاس math

۳- راجع به کاربرد متدهایی که در جدول بالا نوشته‌اید با هم کلاس خود بحث کنید.

۴- خروجی برنامه زیر چیست؟

```
using System;
class Convert
{
    static void Main(string[] args)
    {
        int a, b;
        double d = 0.75;
        a = (int) Math.Round(d);
        b = (int)d;
        Console.WriteLine(a);
        Console.WriteLine(b);
    }
}
```

۵- حاصل عبارت زیر چیست؟

`Math.Truncate(Math.Round(Math.Pow(2,3)/5) + 2.8)`

۶- خروجی دستورات زیر چیست؟

```
static void Main(string[] args)
{
    int [] numbers = { 7, 42, 16, 8, 16, 14 };
    Array.Reverse(numbers);
    Array.Sort(numbers);
    for (int i = 0; i < 6; i++)
        Console.WriteLine(numbers[i]);
    Console.ReadKey();
}
```

۷- در سؤال قبل اگر به جای دستور `for` دستور زیر را می نوشتیم چه چیزی نمایش داده می شد؟

```
Array.IndexOf(numbers, 16);
```

۸- دستورات زیر چه کاری انجام می دهند؟

```
static void Main(string[] args)
{
    string exp = "i am a good student";
    for (int i = 0; i < strings.Length; i++)
        Console.WriteLine(strings[i]);
    Console.ReadKey();
}
```

تمرینات برنامه‌نویسی فصل چهارم

- ۱- می‌خواهیم ۲۰ نفر از دانش‌آموزان را برای نشستن در کارگاه رایانه، به گروه‌های دو نفری تقسیم کنیم. برنامه‌ای بنویسید که نام این ۲۰ نفر را دریافت کند و با مرتب کردن نام آنها هر دو نفر را برای گروه‌های شماره بندی شده از ۱ تا ۱۰ نمایش دهد.
- ۲- برنامه‌ای بنویسید که یک رشته متنی را دریافت کند. کلمات آن را برعکس نمایش دهد. به عنوان مثال اگر رشته متنی "I am a good student" را به برنامه بدهیم متن "student good a am I" را نمایش دهد.
- ۳- می‌خواهیم در کلاس خود یک فعالیت پژوهشی با حضور ۱۰ دانش‌آموز اول لیست کلاسی انجام دهیم. برنامه‌ای بنویسید که اسامی ۳۰ دانش‌آموز را دریافت کرده و اسامی ۱۰ نفر اول دفتر کلاسی (می‌دانید که اسامی دانش‌آموزان در دفتر کلاسی بر اساس حروف الفبا مرتب شده است) را که در دفتر کلاسی پژوهش (آرایه جدید) نیز وارد می‌کند، نمایش دهد.

متن زیر از MSDN با موضوع آرایه برداشت شده است. آن را با کمک هم کلاسی خود ترجمه کنید و به کلاس ارائه نمایید.

Initializing Arrays

C# provides simple and straightforward ways to initialize arrays at declaration time by enclosing the initial values in curly braces ({}). The following example shows the way to initialize an array.

Note If you do not initialize an array at the time of declaration, the array members are automatically initialized to the default initial value for the array type. Also, if you declare the array as a field of a type, it will be set to the default value null when you instantiate the type.

```
int[] numbers = new int[5] {1, 2, 3, 4, 5};  
string[] names = new string[3] {"Matt", "Joanne", "Robert"};
```

واژگان و اصطلاحات انگلیسی فصل چهارم

ردیف	واژه انگلیسی	معنی به فارسی
۱	Area	
۲	Array	
۳	Immutable	
۴	Upper	
۵	Lower	
۶	Radius	

ایجاد برنامه‌های ویندوزی با استفاده از ویژوال استودیو

خروجی تمام برنامه‌هایی که در کتاب برنامه‌سازی یک و همچنین فصل‌های قبلی در این کتاب نوشتید در محیط کنسول قابل مشاهده بود. در این فصل می‌خواهیم با روش برنامه‌نویسی در محیط ویندوز آشنا شویم تا بتوانیم برنامه‌هایی با ظاهری زیباتر، مانند برنامه‌های کاربردی رایج در ویندوز بنویسیم. برای محقق شدن این هدف، در دو فصل آینده برنامه‌های خود را در محیط ویندوزی خواهیم نوشت.

پس از پایان این فصل انتظار می‌رود که فراگیر بتواند:

- ۱- تفاوت بین برنامه‌های کنسولی و ویندوزی را توضیح دهد.
- ۲- واسط گرافیکی کاربر و کاربرد کلاس From را توضیح دهد.
- ۳- گزینه‌های مورد نیاز پنجره‌ها، منوها و نوار ابزارهای IDE را در محیط ویندوزی مورد استفاده قرار دهد.
- ۴- پروژه ویندوزی با یک فرم ایجاد نماید و ویژگی‌های مورد نیاز را تنظیم نماید.
- ۵- کنترل‌های برچسب و جعبه تصویر را به فرم خود اضافه کند و برخی ویژگی‌های آنها را به دلخواه تغییر دهد.
- ۶- با استفاده از VS برنامه ویندوزی را ترجمه و اجرا نماید.

۱-۵- تفاوت برنامه‌های کنسولی و برنامه‌های ویندوزی

برنامه‌های کنسولی که تاکنون نوشته‌اید برای یادگیری مفاهیم اساسی و مقدماتی زبان برنامه‌نویسی C# و به دست آوردن تجربه کدنویسی خوب است اما برای تولید برنامه‌های کاربردی که در زندگی روزمره کاربرد داشته باشد، چندان مناسب نیست. امروزه بیشتر کاربران کامپیوتر، با برنامه‌های کاربردی رایج کار کرده‌اند و توقع دارند برنامه‌ای که شما می‌نویسید دارای ظاهر و عملکردی شبیه برنامه‌های رایج در ویندوز باشند. در اجرای برنامه‌هایی که از این به بعد می‌نویسید انتظار می‌رود که با اجرای آن، یک پنجره باز شود و در داخل آن منوها و ابزارهای مختلفی دیده شود که با کلیک کردن روی هر منو یا روی گزینه‌ای از یک منو، پنجره دیگری ظاهر شود. در این پنجره‌ها، اطلاعاتی از کاربر در قالب یک فرم دریافت می‌شود و یا اطلاعاتی نشان داده می‌شود. در هر فرم، قسمت‌هایی برای ورود اطلاعات و کلیدهایی برای تأیید و ثبت اطلاعات و یا لغو کردن عملیات وجود دارد. به عنوان نمونه در شکل زیر نمونه‌ای از برنامه حدس عدد را که در کتاب برنامه‌سازی ۱ در محیط کنسول ساخته بودید، در محیط ویندوز مشاهده می‌کنید. این برنامه را در ادامه می‌سازید.



شکل ۱-۵- یک فرم و برنامه کاربردی

در برنامه‌های کنسولی، اجرای برنامه از متدی به نام Main() شروع می‌شود و دستورات داخل آن، به ترتیب و خط به خط اجرا می‌شوند. در چنین برنامه‌هایی با استفاده از متدهای موجود مانند ReadLine() برای دریافت یک رشته، درخواست‌هایی به سیستم عامل داده می‌شود که عملیاتی را برای ما انجام دهد.



شکل ۲-۵

در برنامه‌های ویندوزی نیز اجرای برنامه با متدی به نام Main() شروع می‌شود اما برخلاف برنامه‌های کنسول، برنامه در حالت انتظار^۱ قرار می‌گیرد تا یک اتفاق یا رویداد^۲ رخ دهد که در این صورت نسبت به آن واکنش نشان دهد. مثلاً وقتی کاربر با ماوس بر روی دکمه^۳ در شکل ۵-۱ کلیک می‌کند، سیستم عامل که مدیر و کنترل کننده منابع سیستم است متوجه می‌شود و این اتفاق را به عنوان یک رویداد به برنامه اطلاع می‌دهد.

رویداد چیست؟ رویداد یک اطلاع^۴ است که از طرف سیستم عامل به برنامه داده می‌شود تا نشان دهد که یک اتفاق رخ داده است.

برنامه‌های ویندوزی باید به رویدادهایی که برای آنها مهم است، عکس العمل نشان دهند.



شکل ۵-۳

برای مقایسه بهتر برنامه‌های کنسولی و برنامه‌های ویندوزی، مثال زیر را در نظر بگیرید : فرض کنید شما در منزل هستید و قرار است برای شما مهمان بیاید. اگر بخواهید به موقع درب منزل را برای وی باز کنید می‌توانید به یکی از روش‌های زیر عمل کنید :



شکل ۵-۴

۱- هر چند لحظه یک بار، به سمت در منزل بروید و بیرون در را نگاه کنید که آیا مهمان شما آمده است.

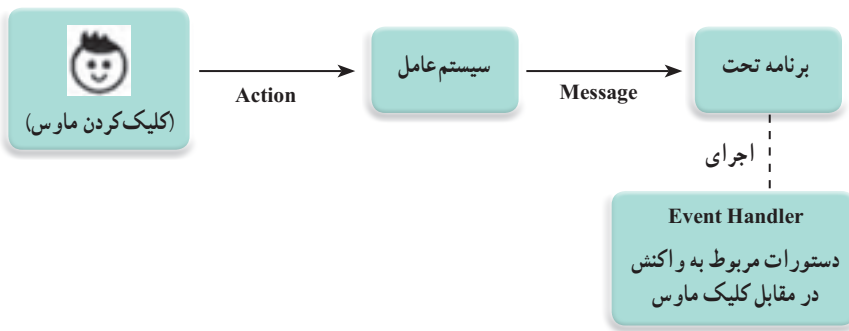
۲- می‌توانید کارهای دیگر خود را رها کنید و در جلوی در منزل خود، منتظر شوید تا وی بیاید.

۳- می‌توانید در داخل منزل مشغول انجام کارهای خود باشید، هر وقت که مهمان آمد، زنگ زد و شما را از ورودش مطلع کرد، به استقبال وی بروید و در را برایش باز کنید.

- ۱- Wait state
- ۲- Event
- ۳- Button
- ۴- Notification

در برنامه‌های کنسولی، در دستوراتی مانند دستور دریافت داده، اجرای برنامه متوقف می‌شود و تا زمانی که کلید Enter زده شود دستورات بعدی اجرا نمی‌شوند. پس از اینکه کاربر اطلاعات را وارد کرد و کلید Enter را زد، خطوط بعدی برنامه به ترتیب اجرا می‌شوند. در مقایسه با مثال بالا می‌توان گفت در برنامه‌های کنسولی از روشی مانند روش ۱ و یا روش ۲ استفاده می‌شود. به نظر شما در برنامه‌های ویندوزی، از کدام روش استفاده می‌شود؟

اما در محیط ویندوز، هنگامی که کاربر عملی^۱ را انجام می‌دهد مثلاً کلیدی از صفحه کلید را فشار می‌دهد و یا با استفاده از ماوس بر روی منویی و یا کلیدی کلیک می‌کند، از طرف سیستم عامل به برنامه اطلاع داده می‌شود که یک رویداد رخ داده است. در واقع برنامه‌های ویندوز با دریافت پیام‌هایی از سیستم عامل، از رخ دادن رویداد مطلع می‌شوند. بنابراین در برنامه‌های ویندوزی شما به عنوان برنامه نویس، باید پیش بینی کنید که اگر کاربر عملی را انجام دهد، برنامه چگونه نسبت به آن واکنش نشان دهد و برای این منظور باید متدهایی را بنویسید که در مواجهه با یک رخداد یا رویداد مانند کلیک ماوس در یک محل، به کامپیوتر اعلام کند که چه کاری باید انجام شود. به این متدها، مانند Event Handler گفته می‌شود که برای سادگی در این کتاب به جای آن به اختصار EH را به کار می‌بریم.



شکل ۵-۵

به این ترتیب هنگامی که کاربر بر روی کلیدی با ماوس کلیک می‌کند، این رویداد از طریق سیستم عامل شناسایی می‌شود و از طرف سیستم عامل یک پیام به برنامه فرستاده می‌شود و متد EH مربوط به آن رویداد که قبلاً نوشته‌ایم به طور خودکار اجرا می‌شود. طبیعی است که اگر برنامه فاقد

۱- Action

۲- Message

متدی برای یک رویداد باشد، هر چقدر که سیستم عامل پیام بفرستد برنامه در مقابل آن واکنشی نشان نخواهد داد. شکل کلی یک متد EH به صورت زیر است:

```
(جزئیات رویداد، فرستنده پیام) نام متد void
{
    دستورات واکنش به رویداد
}
```

شکل ۵-۶

هر متد EH دارای دو پارامتر ورودی است که اولی نوع شیء فرستنده پیام را مشخص می‌کند و پارامتر دوم مربوط به جزئیات رویداد است مثلاً در مورد کلیک ماوس شامل اطلاعاتی در مورد موقعیت مکانی ماوس در لحظه کلیک و کلیدی (کلید چپ، راست یا وسط) از ماوس است که فشار داده شده است.

هنگامی که یک برنامه ویندوزی مانند برنامه ماشین حساب را اجرا می‌کنید، این برنامه پس از نمایش پنجره خود، در حالت انتظار قرار می‌گیرد، تا شما مثلاً بر روی دکمه ای کلیک کنید و یا از طریق صفحه کلید، علامت یا رقمی را وارد کنید که در این صورت یک رویداد رخ می‌دهد و EH مربوطه اجرا می‌شود.

یک تفاوت دیگر بین برنامه ویندوزی و کنسولی این است که، برنامه‌های ویندوزی پس از واکنش به رویدادها و انجام عملیات مربوطه، مجدداً در حالت انتظار برای رویداد بعدی به سر می‌برند تا در نهایت کاربر از برنامه خارج گردد.

در ضمن با توجه به اینکه رویدادهای مختلفی ممکن است از طریق کاربر رخ دهد که قابل پیش‌بینی نیست، بنابراین با اجرای برنامه، مشخص نیست که کدام رویداد ممکن است ابتدا رخ دهد و به عبارت دیگر ترتیبی برای رویدادها، قابل پیش‌بینی نیست.

۲-۵- واسط گرافیکی کاربر

در طول دوره‌های مختلف تحصیلی، ممکن است، یک روزنامه دیواری با دوستان خود درست کرده باشید. برای ایجاد روزنامه دیواری ابتدا به یک مقوا نیاز دارید تا مطالب خود را بر روی آن

بنویسید و یا عکس‌ها و بریده‌های مجلات را بر روی آن بچسبانید. در ساخت یک برنامه ویندوزی نیز باید صفحه یا فرمی در اختیار داشته باشید تا انواع دکمه‌ها، منوها، تصاویر و نوشته‌ها را روی آن قرار دهید. این فرم، واسط گرافیکی کاربر یا GUI نامیده می‌شود. فرم، صفحه‌ای است که اجزای گرافیکی گوناگونی بر روی آن قرار می‌گیرد. اندازه یک فرم متناسب با تعداد و اندازه اشیای گرافیکی است که قرار است روی آن جای گیرند (شکل ۵-۷).

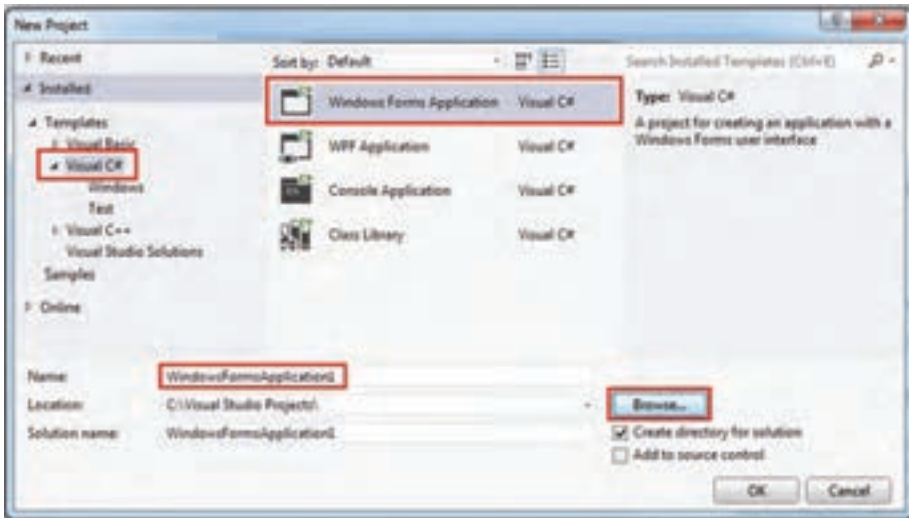


شکل ۵-۷- یک فرم خالی مانند یک مقوای روزنامه دیواری

همان طور که برای خرید مقوای روزنامه دیواری، لازم است به یک فروشگاه لوازم التحریر مراجعه کنید، در زبان برنامه نویسی C# نیز، برای ایجاد یک فرم باید به کتابخانه ارزشمند Net. مراجعه کنید. در این کتابخانه، کلاسی به نام Form تعریف شده است که برای ایجاد یک فرم لازم است از آن به عنوان پایه و مبنای کار خود استفاده کنیم.

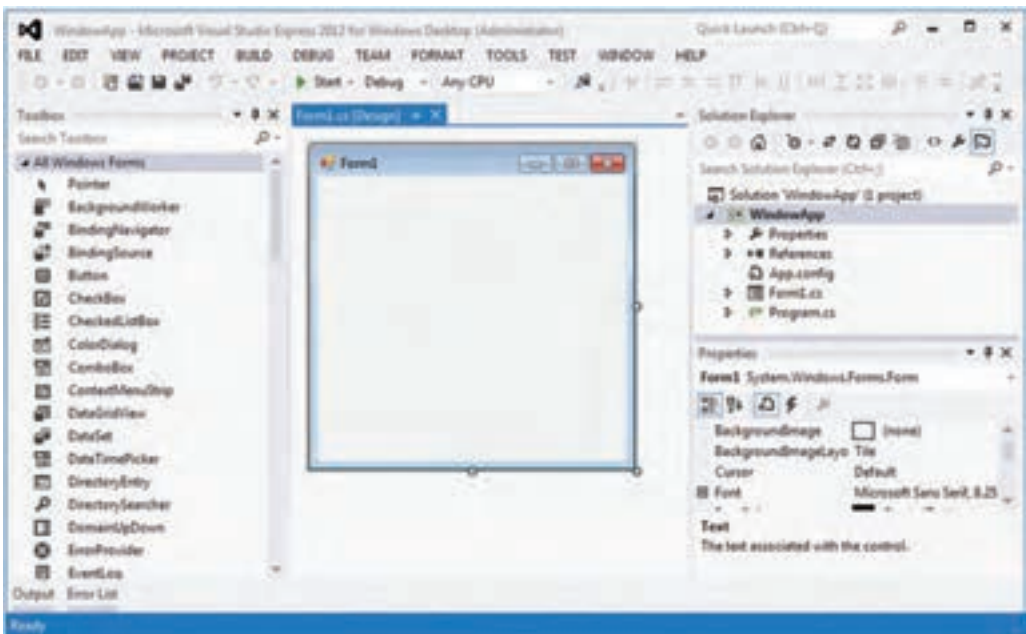
۳-۵- ایجاد یک پروژه ویندوزی با کمک VS

برای ایجاد یک برنامه ویندوزی، VS را اجرا کنید و در صفحه آغازین، روی گزینه New Project کلیک کنید در پنجره‌ای که مطابق شکل ۵-۸ نمایان می‌شود، گزینه Visual C# و سپس Windows Forms Application را انتخاب کنید.



شکل ۸-۵ پنجره ایجاد پروژه جدید

می‌توانید نام و مسیر ذخیره سازی پروژه را مانند برنامه‌های کنسولی، انتخاب کنید. پس از تأیید و کلیک بر روی دکمه OK، وارد محیط IDE (شکل ۹-۵) می‌شوید.



شکل ۹-۵ پنجره IDE پروژه ویندوزی

همان‌طور که در شکل ۵-۹ مشاهده می‌شود خیلی با این محیط غریبه نیستیم و شبیه محیط تولید برنامه کنسولی است. در اینجا گذری بر روی پنجره‌های مختلف آن خواهیم داشت.

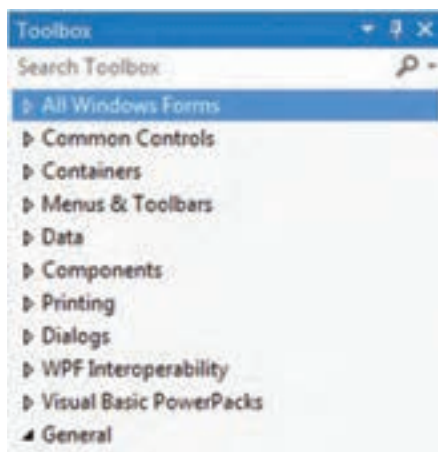
۵-۳-۱- آشنایی با پنجره‌های IDE

در بالای صفحه، منوها و نوار ابزارها^۱ مانند محیط تولید برنامه‌های کنسولی وجود دارد. البته همان‌طور که در شکل ۵-۱۰ دیده می‌شود یک منو به نام Format اضافه شده است که از گزینه‌های آن برای شکل‌دهی ظاهری و تنظیم جای‌دهی کنترل‌های یک فرم استفاده می‌شود.



شکل ۵-۱۰- نوار منوها و نوار ابزارهای IDE در یک پروژه ویندوزی

در سمت چپ صفحه (IDE)، پنجره‌ای به نام جعبه ابزار^۲ دیده می‌شود. در پنجره جعبه ابزار، نام و آیکن کنترل‌های قابل استفاده، لیست شده است. کنترل‌ها در جعبه ابزار، دسته‌بندی شده‌اند و هنگامی که بر روی نام یک گروه، کلیک می‌کنید کنترل‌های مربوطه نمایش داده می‌شوند و اگر مجدداً روی نام آن گروه کلیک کنید، لیست مربوطه بسته می‌شود. در شکل ۵-۱۱ جعبه ابزار به صورت گروه‌بندی نشان داده شده است.



شکل ۵-۱۱- پنجره جعبه ابزار

۱- Tool bar

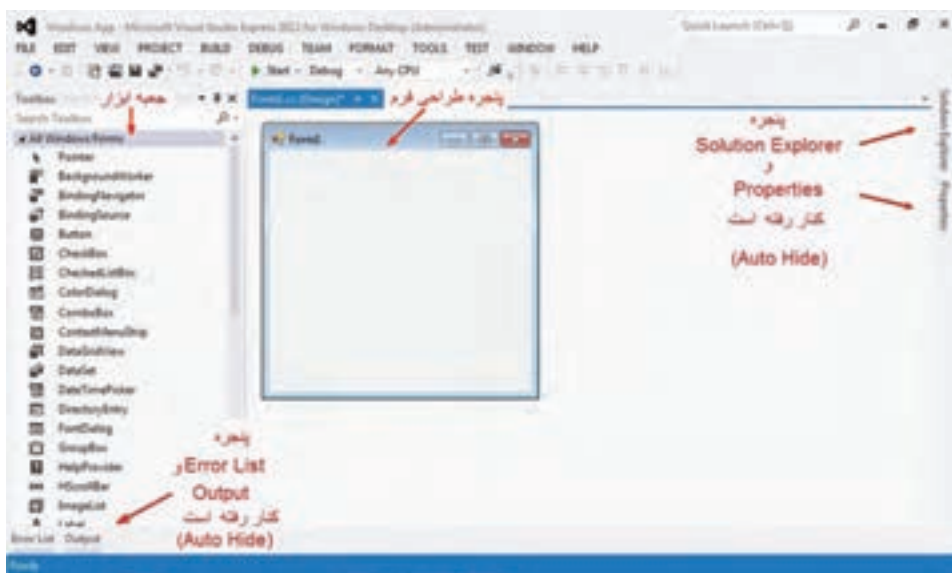
۲- Tool box



شکل ۵-۱۲- پنجره طراحی فرم

در وسط صفحه، یک فرم دیده می‌شود که از آن برای طراحی واسط کاربری و قراردادن کنترل‌ها مانند مقوای روزنامه دیواری، استفاده خواهیم کرد. اندازه طول و عرض فرم به راحتی با استفاده از ماوس و با کشیدن گوشه‌های آن، که در شکل ۵-۱۲ مشخص شده است، قابل تغییر است. با این پنجره که پنجره طراحی فرم نامیده می‌شود، زیاد کار داریم.

با توجه به اینکه محیط VS قابل تنظیم است، ممکن است صفحه‌ای که روی کامپیوتر شما دیده می‌شود با آنچه که در شکل ۵-۹ مشاهده می‌کنید، کمی متفاوت باشد. مثلاً در شکل ۵-۱۳، پنجره‌هایی در حالت Auto Hide، قرار دارند و با کلیک بر روی نام آنها، پنجره دیده می‌شود و با کلیک در نقطه دیگری از صفحه، پنجره‌ها به طور خودکار کنار می‌روند و فقط نام پنجره‌ها دیده می‌شوند.



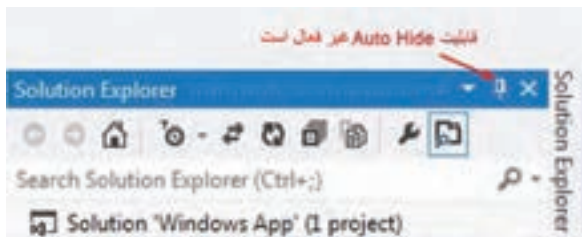
شکل ۵-۱۳- حالت Auto Hide پنجره‌ها

از قابلیت Auto Hide برای کنار رفتن موقتی پنجره‌ها و برای بزرگ‌شدن محیط کار استفاده می‌شود. برای اینکه پنجره‌ای را از حالت Auto Hide خارج کنید و یا این قابلیت را به آن بدهید، کافی است در روی خط عنوان پنجره بر روی آیکن پوزر^۱ کلیک کنید. شکل ۱۴-۵ را مشاهده کنید.

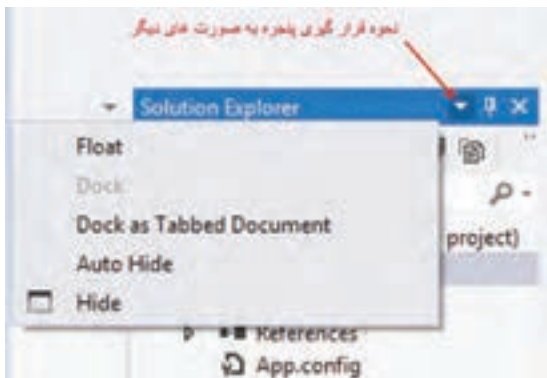


شکل ۱۴-۵- قابلیت کنار رفتن پنجره به طور خودکار فعال است.

اگر پوزر در حالت افقی مانند شکل ۱۴-۵ باشد، قابلیت Auto Hide فعال است و اگر روی آن کلیک کنید، پوزر به حالت عمودی قرار می‌گیرد و پنجره ثابت می‌ماند. شکل ۱۵-۵ را مشاهده کنید.



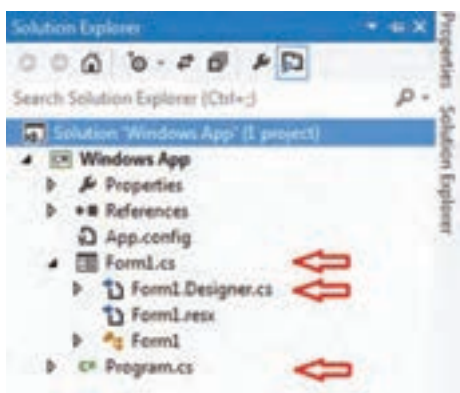
شکل ۱۵-۵- حالت کنار رفتن خودکار پنجره غیر فعال است.



شکل ۱۶-۵- نحوه قرار گیری و نمایش پنجره

در کنار آیکن پوزر، آیکن مثلثی شکلی مانند شکل ۱۶-۵ نیز قرار دارد که با کلیک بر آن منوی Windows Position باز می‌شود تا روش‌های دیگری برای قرارگیری پنجره فراهم کند. در قسمت کار در کارگاه آنها را تجربه خواهید کرد.

^۱ - Pushpin

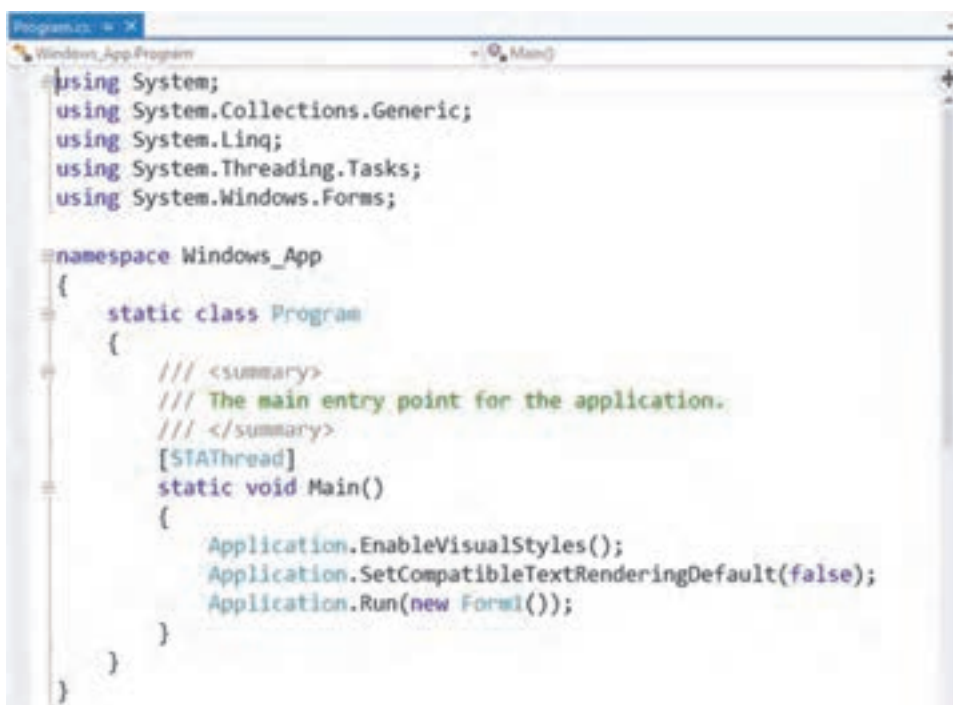


شکل ۱۷-۵ پنجره Solution Explorer در یک پروژه ویندوزی

به پنجره Solution Explorer مطابق شکل ۱۷-۵ دقت کنید در این پنجره، نسبت به آنچه که در برنامه کنسولی مشاهده کردید، موارد بیشتری دیده می‌شود.

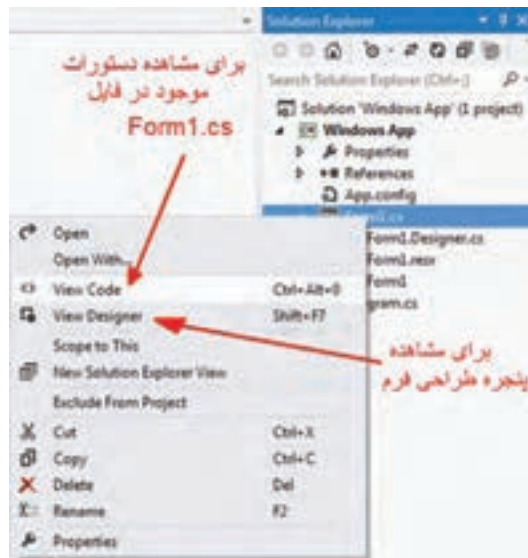
علاوه بر فایل Program.cs که نام آن برای شما آشنا است، فایل‌های دیگری از جمله دو فایل Form1.Designer.cs و Form1.cs به نام‌های دیده می‌شوند. دستورات یک برنامه ویندوزی در این سه فایل قرار می‌گیرد. اگر در پنجره Solution Explorer روی نام فایل Program.cs،

دوبار کلیک کنید، پنجره‌ای باز می‌شود و محتویات آن را نشان می‌دهد. در این فایل کلاس اصلی برنامه و متد Main() مطابق شکل ۱۸-۵ قرار دارد.



شکل ۱۸-۵ محتوای فایل Program.cs

اگر در پنجره Solution Explorer بر روی کلمه Form1.cs کلیک راست کنید، منوی مانند شکل ۱۹-۵ ظاهر می‌گردد که شامل گزینه‌هایی برای انجام عملیات مختلف بر روی آن فایل است. به این منوها که گزینه‌های آن مربوط به یک موضوع خاص است منوی موضوعی^۱ یا محلی گفته می‌شود.



شکل ۱۹-۵- منوی موضوعی در مورد فایل Form1.cs

در منوی ظاهر شده، اگر بر روی گزینه View Designer کلیک کنید یا از میان‌بر^۲ آن Shift+F7 استفاده کنید، پنجره طراحی فرم را خواهید دید.

نکته

در هنگام برنامه‌نویسی در محیط VS، سعی کنید از کلیدهای میان‌بر استفاده کنید تا سرعت شما در انجام عملیات بالا رود. مثلاً هنگامی که در پنجره Solution Explorer فایل Form1.cs انتخاب شده است، برای مشاهده دستورات آن و یا برای مشاهده پنجره طراحی فرم که بارها مورد استفاده قرار می‌گیرد، از میان‌برهای زیر استفاده کنید.

Ctrl + Alt + 0 مشاهده دستورات فایل فرم

Shift + F7 مشاهده پنجره طراحی فرم

^۱ Context menu

^۲ Shortcut

در منوی موضوعی فایل Form1.cs، اگر بر روی گزینه View Code کلیک کنید، محتوای فایل Form1.cs را مانند شکل ۵-۲۰ مشاهده خواهید کرد، که قسمتی از تعریف کلاس Form1 است.



```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Windows_App
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}

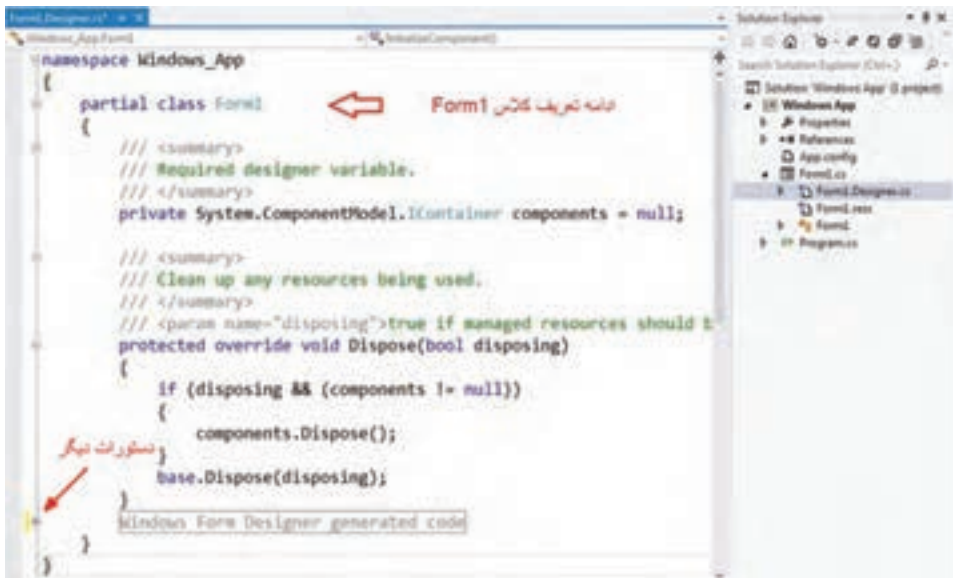
```

شکل ۵-۲۰- محتوای دستورات و برنامه فایل Form1.cs

به خط عنوان تعریف کلاس Form1 توجه کنید، کلمه کلیدی partial در خط عنوان کلاس نشان می‌دهد که بخشی از تعریف کلاس Form1 در فایل دیگری قرار دارد. قسمت دیگر تعریف کلاس Form1 کجا است؟

فایل دیگری که VS ایجاد می‌کند و شما نباید محتویات آن را دستکاری کنید، فایل Form1.Designer.cs است. هنگامی که از جعبه ابزار سمت چپ صفحه، کنترلی را انتخاب و آن را در روی فرم قرار می‌دهید و ویژگی‌های آن را مطابق سلیقه خود تنظیم می‌کنید، VS تنظیمات شما را به دستورات و کدهای زبان C# تبدیل می‌کند. این دستورات به عنوان بخشی از تعریف کلاس Form1 به وسیلهٔ VS در فایل Form1.Designer.cs به‌طور خودکار نوشته می‌شود و کار شما را بسیار آسان می‌کند. بنابراین بقیه تعریف کلاس Form1 در این فایل قرار دارد. در پنجره Solution Explorer آن را پیدا کرده و با دو بار کلیک، با احتیاط آن را باز می‌کنیم. همان‌طور که در شکل ۵-۲۱ مشاهده می‌کنید ادامه تعریف کلاس Form1 دیده می‌شود. دستورات و محتویات این فایل، با

قرار دادن کنترل‌ها بر روی فرم، افزایش می‌یابد. برای جلوگیری از دستکاری دستورات این فایل و در دسترس قرار ندادن آنها، مترجم آنها را در لحظه اول نمایش نمی‌دهد.



شکل ۲۱-۵- محتوای دستورات و برنامه فایل Form1.Designer.cs

برای نمایش دستورات، در سمت چپ پنجره‌ای که محتوای فایل Designer را نشان می‌دهد، یک خط عمودی را مشاهده می‌کنید که در بین آن علامت‌های - و در بعضی قسمت‌ها علامت + قرار دارد. اگر بر روی علامت + کلیک کنید، منطقه‌ای نمایان می‌شود که تعدادی از دستورات در آن قرار گرفته است.^۱ لطفاً این دستورات را بدون آگاهی تغییر ندهید که در این صورت قسمت طراحی فرم دچار مشکل می‌شود.

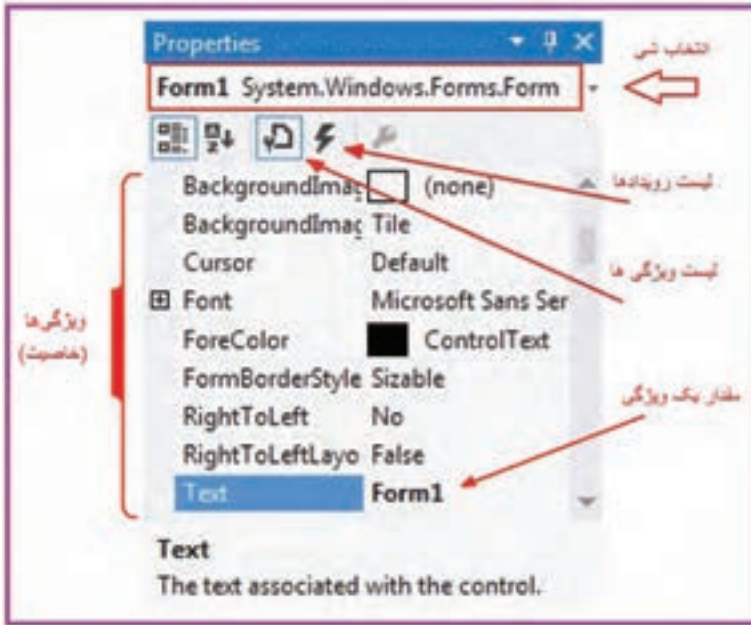
۲-۳-۵- پنجره ویژگی‌ها و خواص اشیاء

پنجره‌ای که در برنامه‌های ویندوزی از آن زیاد استفاده خواهیم کرد، پنجره Properties است که در شکل ۲۲-۵ مشاهده می‌کنید.

۱- با استفاده از راهنمای #region و #endregion، این امکان مهیا می‌شود که تعدادی دستور در لحظه اول نشان داده نشوند و در صورت لزوم با کلیک بر روی علامت +، نمایان شوند.

Formatted: Font: 11 pt, Complex Script Font: 11pt

Formatted: Font: 11 pt, Complex Script Font: 11pt



شکل ۲۲-۵ - پنجره Properties

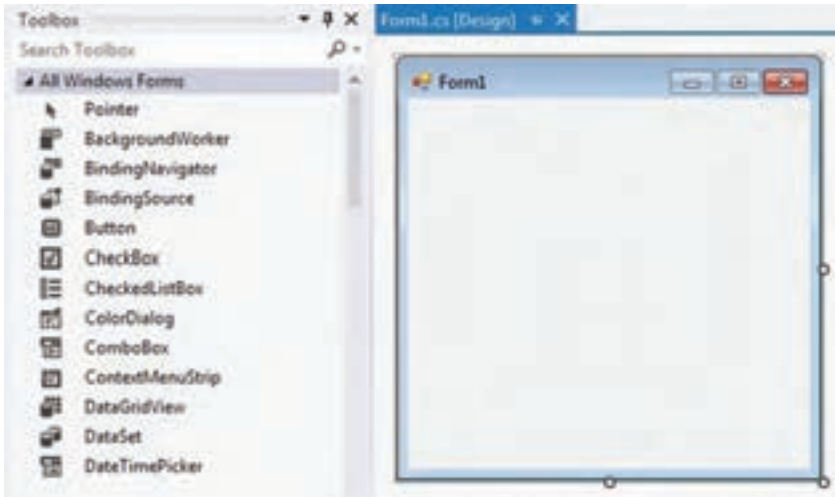
در این پنجره ویژگی‌ها و خواص شی و همچنین واکنش به رویدادهای هر شی را می‌توانیم مقاردهی و تنظیم کنیم. در قسمت بالای این پنجره، نام یک شی که در فرم انتخاب شده، نشان داده شده است و از طریق مثلث کوچک کنار آن می‌توان شی دیگری را از بین اشیای موجود روی فرم انتخاب کنیم. در زیر نام شی، می‌توان با کلیک بر روی آیکنی که نظیر رعد و برق است، لیست رویدادهای شی مشاهده و یا بر روی آیکن کنار آن کلیک کرد تا لیست ویژگی‌های شی را مشاهده شود. در هر کدام از انتخاب‌ها، لیست نشان داده شده را می‌توان به ترتیب نام و یا به ترتیب عملکرد دسته‌بندی شده است، مشاهده نماید. در شکل ۲۳-۵، آیکن‌های مربوطه دیده می‌شوند.



شکل ۲۳-۵ - آیکن‌های ترتیب نمایش ویژگی‌ها در پنجره Properties

۳-۳-۵- استفاده از جعبه ابزار و قراردادن کنترل‌ها در روی فرم

همان‌طور که در فصل قبل بیان شد، یک برنامه ساده ویندوزی از یک فرم تشکیل می‌شود که به عنوان نگهدارنده^۱ برای قرار گرفتن اجزای گرافیکی دیگر، استفاده می‌شود. شکل ۲۴-۵ را مشاهده کنید.



شکل ۲۴-۵- پنجره طراحی و جعبه ابزار

در حالی که پنجره طراحی فرم را مشاهده می‌کنید، می‌توانید از جعبه ابزار استفاده کرده و کنترل مورد نظر خود را پیدا کنید. برای قرار دادن یک کنترل بر روی فرم، می‌توانید روی کنترل مورد نظر دوبار کلیک کنید و یا با درگ کردن کنترل و رها کردن آن روی فرم، کنترل مورد نظر را روی فرم بچسبانید. البته بعضی از کنترل‌ها مانند کنترل زمان‌سنج (تایمر)^۲، با توجه به عملکردشان، ممکن است در قسمت پایین فرم قرار گیرند که با آنها در این کتاب آشنا می‌شوید.

اندازه فرم که برای طراحی استفاده می‌کنید قابل تغییر است. توجه داشته باشید که اگر در پنجره Form Designer، در روی فرم دوبار کلیک کنید، دستوراتی به فایل Form1.cs اضافه می‌شود که مربوط به الگوی، یک EH است که در شکل ۲۵-۵ نشان داده شده است، به دلیل اینکه به طور اتفاقی و اشتباهاً دوبار کلیک کرده‌اید، لذا دستورات اضافه شده را حذف کنید و یا با احتیاط، عمل انجام شده را لغو و یا بی‌اثر کنید.

۱- Container

۲- Timer Control

```

Form1.cs* * X Form1.cs [Design]*
Windows_App.Form1 - Form1_Load(object sender, EventArgs e)

namespace Windows_App
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

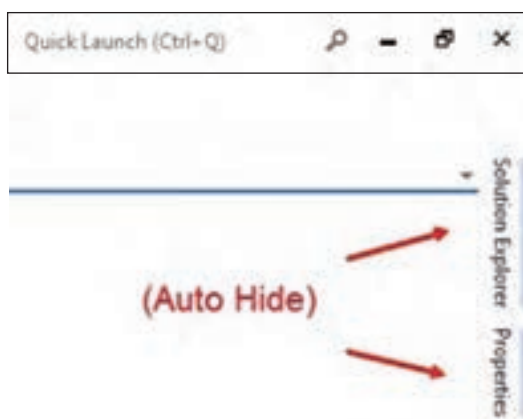
        private void Form1_Load(object sender, EventArgs e)
        {
        }
    }
}

```

شکل ۲۵-۵- دستورات اضافه شده در اثر دو بار کلیک بر روی فرم

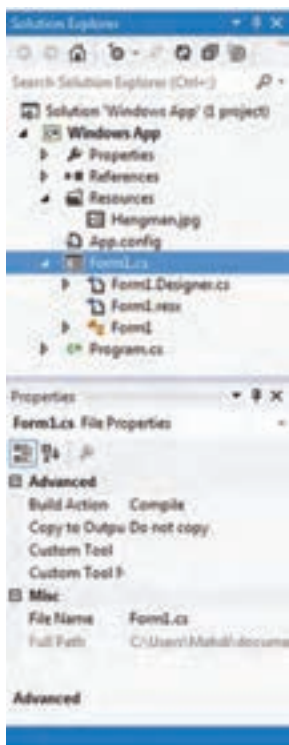
کار در کارگاه ۱: جابه‌جایی پنجره‌ها در VS

- ۱- برنامه VS را اجرا کنید و یک پروژه Windows Forms Application ایجاد کنید.
- ۲- پنجره Solution Explorer را در حالت AutoHide قرار دهید.
- ۳- پنجره Properties را نیز در حالت AutoHide مانند شکل ۲۶-۵ قرار دهید.



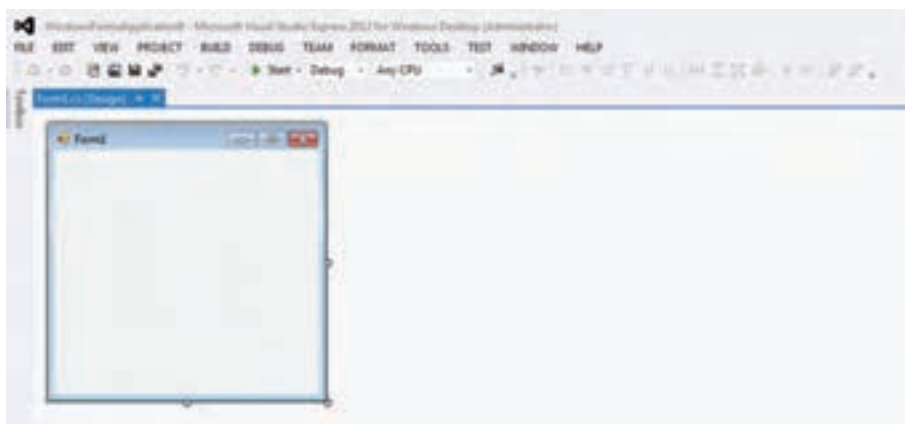
شکل ۲۶-۵- حالت کنار رفتن خودکار

۴- هر دو پنجره را از حالت Auto Hide خارج کرده و به حالت چسبیده^۱ مانند شکل ۵-۲۷ قرار دهید.



شکل ۵-۲۷- پنجره‌ها در حالت چسبیده (Dock)

۵- پنجره Tool box را در حالت Auto Hide قرار دهید تا قسمت طراحی فرم مانند شکل ۵-۲۸ بزرگ‌تر شود. اگر از قبل در حالت Auto Hide است آن را به حالت چسبیده تغییر دهید.



شکل ۵-۲۸- حالت کنار رفتن خودکار پنجره جعبه ابزار

^۱ Dock

۶- ماوس را روی خط عنوان پنجره Solution Explorer برده و با استفاده از درگ کردن، آن را به نقاط دیگر ببرید. توجه کنید که VS مکان‌هایی را مانند شکل ۲۹-۵ به شما پیشنهاد می‌دهد. از آنها استفاده و موقعیت‌های مختلف را تجربه کنید. سعی کنید که مفهوم موقعیت‌های پیشنهادی را بفهمید.



شکل ۲۹-۵- انتقال پنجره به نقاط دیگر صفحه

۷- برای اینکه هر دو پنجره Solution Explorer و Properties از کل فضای سمت راست صفحه، به طور مشترک استفاده نمایند، آنها را به صورت Tab Docked مانند شکل ۳۰-۵ قرار دهید. برای این منظور پنجره Properties را درگ کرده و وقتی روی خط عنوان پنجره Solution Explorer رسید، آن را رها کنید.

۸- مجدداً هر دو پنجره را به صورت چسبیده، مانند شکل ۲۷-۵ قرار دهید.



شکل ۳۰-۵- پنجره‌ها در حالت Tab Docked

اکنون که با پنجره‌های محیط طراحی برنامه ویندوزی در VS آشنا شدیم، می‌توانیم یک نمونه پروژه ویندوزی را قدم به قدم دنبال کنیم.

۴-۵- ساخت یک واسطه کاربری با استفاده از قابلیت طراحی تصویری در VS

هنگامی که از امکانات VS برای طراحی واسطه کاربری استفاده می‌کنید، بسیاری از دستورات و کدهای لازم، بدون اینکه شما متوجه شوید، به طور خودکار به وسیلهٔ VS تولید شده و در قالب فایل‌هایی که در ابتدای این فصل معرفی شد، به برنامه اضافه می‌شوند.

در قسمت کار در کارگاه ۲، مراحل ساخت یک نمونه فرم ساده را بدون نوشتن حتی یک خط برنامه، دنبال می‌کنیم. بدیهی است برای تولید فرم‌های کاربردی و واکنش به رویدادها، علاوه بر استفاده از ابزارها و امکانات VS، باید چند خط برنامه نیز بنویسیم.

کار در کارگاه ۲: ساخت اولین فرم



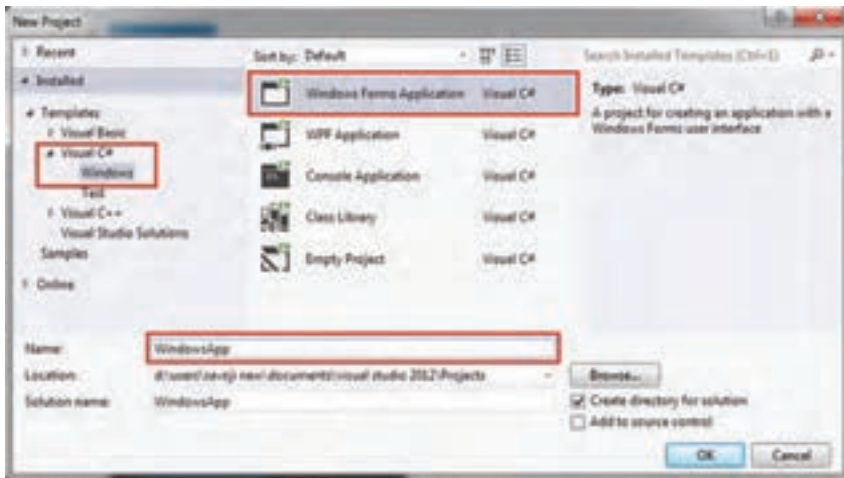
ساخت یک برنامه یا یک فرم مانند

شکل ۳۱-۵

شکل ۳۱-۵- اولین برنامه در ویژوال C#

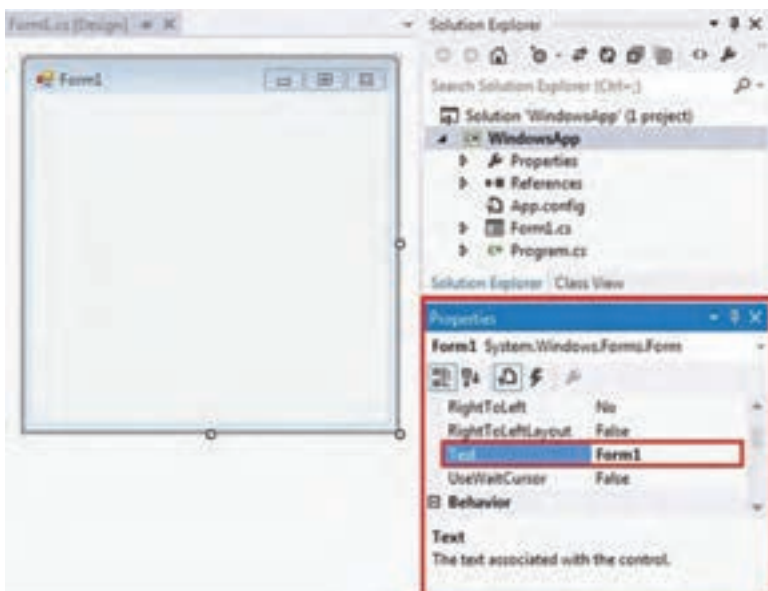
الگوریتم یا روش انجام کار: برای ایجاد چنین فرمی، مراحل زیر را به دقت انجام دهید:

۱- **ایجاد یک پروژه ویندوزی:** وارد برنامه VS شوید و یک پروژه جدید از نوع Windows Forms Application با نام پروژه WindowsApp و در مسیر مشخصی مانند شکل ۳۲-۵ بسازید. اگر در حال حاضر در داخل VS هستید از طریق منوی File، انتخاب گزینه Close Solution، پروژه فعلی را ببندید و سپس یک پروژه جدید مطابق با آنچه که گفته شد، بسازید.



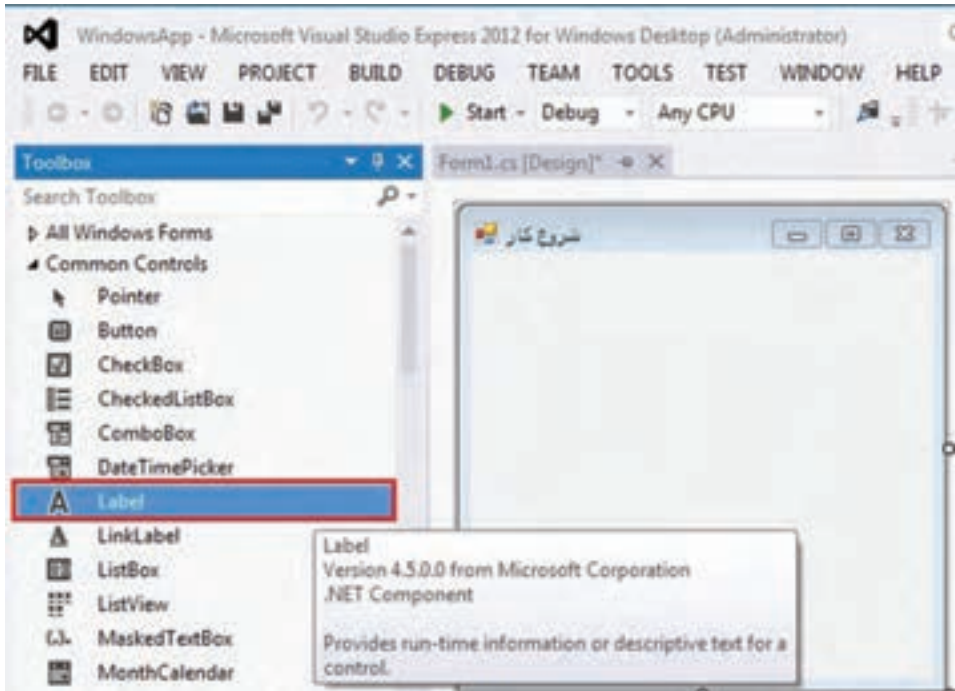
شکل ۳۲-۵- ایجاد پروژه جدید

۲- تعیین خط عنوان فرم: اگر به خط عنوان فرم در شکل ۳۱-۵ توجه کنید، عبارت فارسی «شروع کار» دیده می‌شود. برای تغییر عنوان فرم به عبارت خواسته شده، به پنجره ویژگی‌ها (Properties) رفته و مانند شکل ۳۳-۵ خاصیت Text را پیدا کرده و متن داخل آن یعنی «Form1» را پاک کرده و عبارت جدید «شروع کار» را تایپ کنید. اگر پنجره Properties دیده نمی‌شود از منوی View، بر روی گزینه Properties Window کلیک کنید و یا از میان بر مربوطه استفاده نمایید.



شکل ۳۳-۵- تغییر خاصیت Text مربوط به فرم از طریق پنجره Properties

۳- اضافه کردن یک متن نمایشی بر روی فرم: برای نمایش یک متن یا عبارت بر روی فرم، از کنترل برجسب (Label) استفاده می‌کنیم. به سراغ پنجره جعبه ابزار رفته و کنترل برجسب (Label) را پیدا کرده و آن را بر روی فرم قرار می‌دهیم. شکل ۳۴-۵ را مشاهده کنید.



شکل ۳۴-۵- کنترل برجسب در جعبه ابزار

با یکی از روش‌های زیر می‌توانید کنترل‌ها را بر روی فرم قرار دهید:

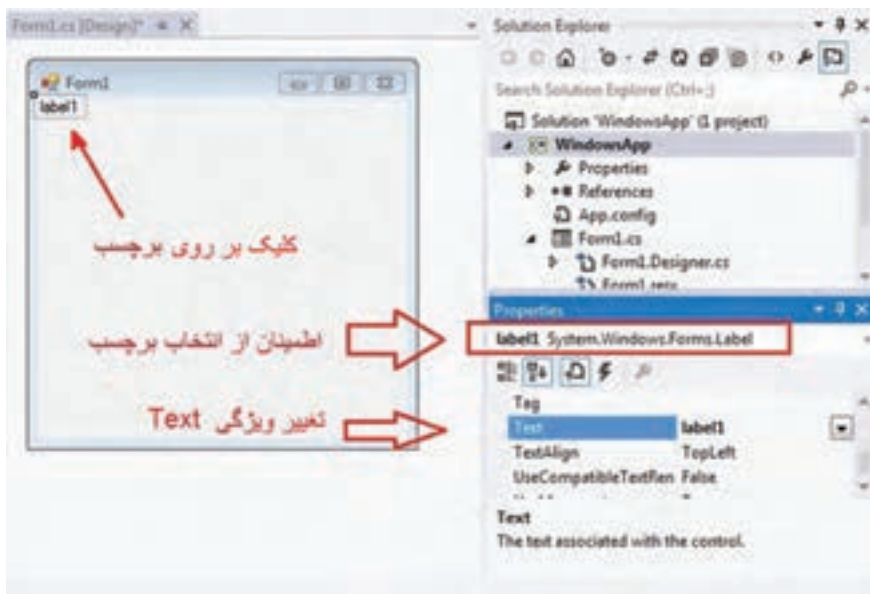
۱- با استفاده از دابل کلیک بر روی کنترل

۲- با استفاده از روش Drag & Drop

۴- تعیین ویژگی *Text* برای برجسب: به پنجره طراحی فرم مراجعه کرده، بر روی برجسب یک بار کلیک کنید تا انتخاب شود. با انتخاب کنترل برجسب روی فرم، پنجره Properties، ویژگی‌های آن را نشان می‌دهد. نام این کنترل به طور خودکار به وسیلهٔ VS انتخاب می‌شود که label1 است. در واقع این نام، نام شیء است که از کلاس Label ایجاد و یا نمونه‌سازی شده است. در خط

اول پنجره Properties، نام شیء به همراه نام کلاس آن، نشان داده شده است.
(Label System.Windows.Forms.Label)

حال در پنجره Properties، پس از اطمینان از اینکه برچسب انتخاب شده، خاصیت Text را مانند شکل ۳۵-۵ پیدا کنید.

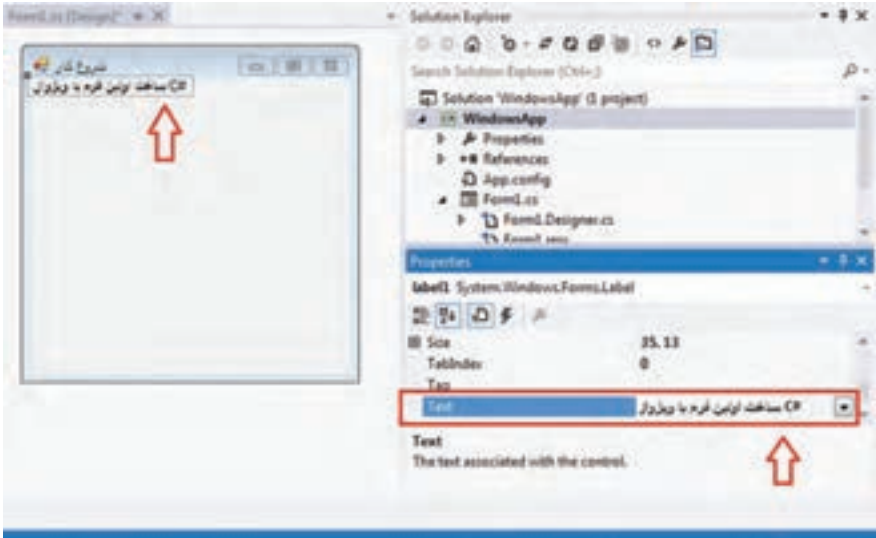


شکل ۳۵-۵- تغییر ویژگی متن کنترل برچسب

برای اطمینان از اینکه برچسب انتخاب شده است به قسمت بالای پنجره Properties توجه کنید، باید نام برچسب را مشاهده کنید. اگر چنین نیست روی فلش کوچک کنار آن کلیک کنید، نام برچسب را پیدا کرده و آن را انتخاب کنید.

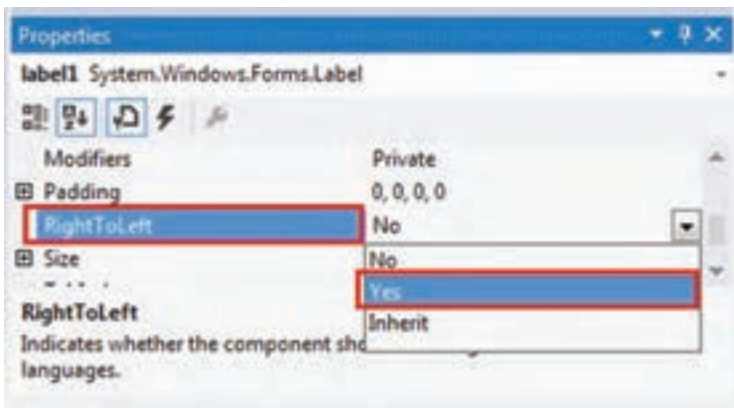
پس از انتخاب کنترل برچسب، عبارت «ساخت اولین فرم با ویژوال C#» را در قسمت Text برچسب وارد کنید.

همان طور که در شکل ۳۶-۵ مشاهده می شود، عبارت مورد نظر در خاصیت Text برچسب، وارد شده اما به هم ریخته است. چون عبارت وارد شده، ترکیبی از کلمات فارسی و همچنین حروف انگلیسی است، لذا مشکل به هم ریختگی کلمات پیش آمده و عبارت «C#» علاوه بر اینکه در جای خود (انتهای جمله) قرار ندارد به شکل برعکس یعنی «#C» دیده می شود. برای رفع این مشکل چه باید کرد؟



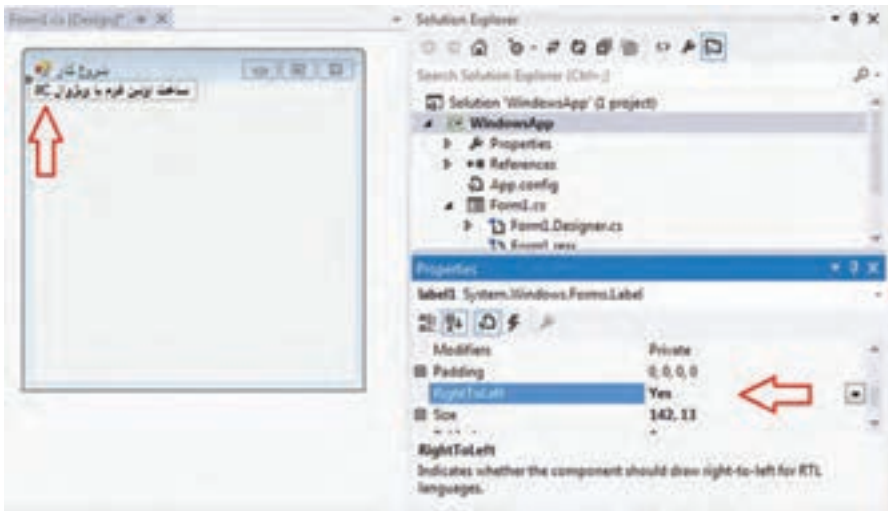
شکل ۳۶-۵ - متن شامل کلمات فارسی و انگلیسی

قبلاً با چنین مشکلاتی در برنامه‌هایی نظیر Word روبه‌رو شده بودید. در اغلب برنامه‌های کاربردی، از زبان‌های محاوره‌ای پشتیبانی می‌شود که در آنها کلمات و جملات از سمت راست به چپ نوشته می‌شود. در .NET نیز چنین است. در کنترل‌ها، خاصیتی به نام RightToLeft برای پشتیبانی از زبان‌های راست به چپ (RTL)، پیش‌بینی شده است. بنابراین به سراغ خاصیت RightToLeft بروید و مقدار آن را به Yes تغییر دهید. شکل ۳۷-۵ را ملاحظه کنید.



شکل ۳۷-۵ - ویژگی RightToLeft برای پشتیبانی از زبان‌های RTL

با تغییر دادن مقدار ویژگی RightToLeft، به مقدار Yes، عبارت نوشته شده در برچسب، به صورت شکل ۵-۳۸ خواهد شد:



شکل ۵-۳۸- تغییر ویژگی RightToLeft

همان طور که در شکل ۵-۳۸ مشاهده می کنید، تنها اشکال باقیمانده، برعکس بودن عبارت "C#" است که برای تصحیح آن به خاصیت Text بروید و آن را به صورت "#C" تایپ کنید.

۵- تعیین ویژگی های فونت، رنگ و محل قرارگیری برچسب: برای تعیین نوع فونت

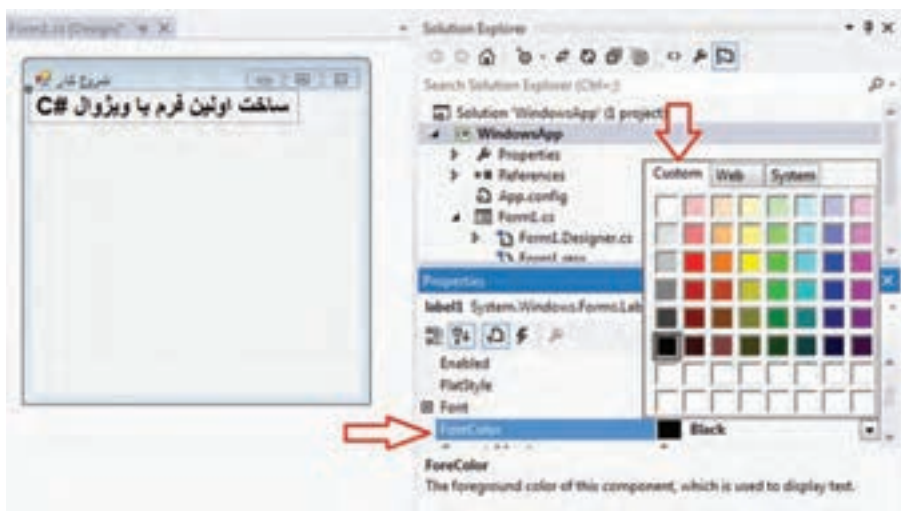


شکل ۵-۳۹- پنجره فونت

کنترل برچسب، از خاصیت Font در پنجره Properties استفاده کنید و روی علامت سه نقطه کلیک کنید تا لیست فونت های موجود در کامپیوتر ظاهر شود. پنجره ای مانند شکل ۵-۳۹ مشاهده می شود که در آن می توانید، فونت دلخواه را به همراه اندازه و سبک آن انتخاب کنید. سعی کنید، فونتی را انتخاب کنید که نوشته های فارسی خوب

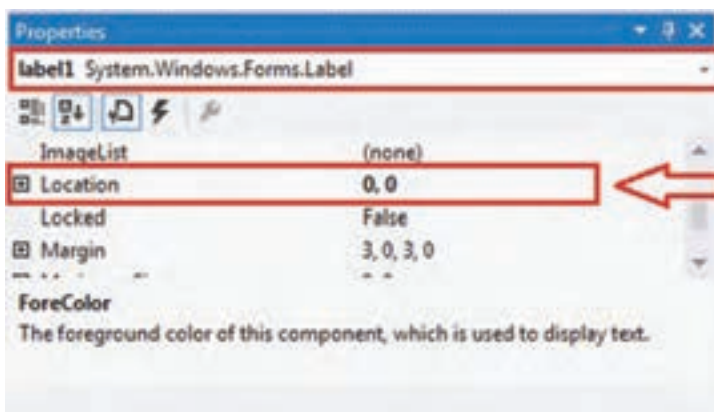
دیده شوند.

همچنین برای تعیین رنگ نوشته از خاصیت ForeColor استفاده می‌کنیم. پس از کلیک بر روی علامت فلش، پنجره ای برای تعیین رنگ باز می‌شود، در آن پنجره مطابق شکل ۴۰-۵، بر روی سربرگ Custom کلیک کنید.



شکل ۴۰-۵ - پنجره رنگ

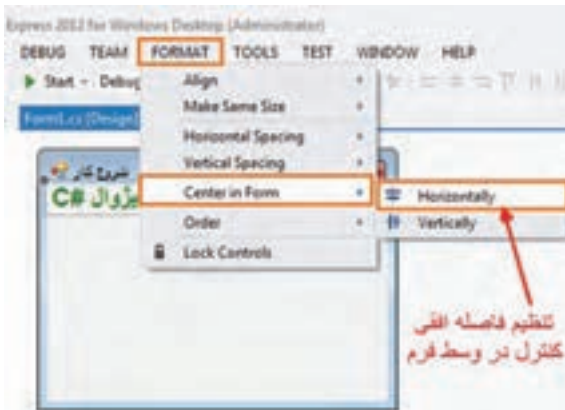
حال می‌خواهیم محل قرارگیری کنترل را تنظیم کنیم. قبل از انجام این کار که به سادگی با استفاده از ماوس می‌توانید انجام دهید، به پنجره Properties مراجعه کنید و ویژگی Location را مطابق شکل ۴۱-۵ پیدا کرده، روی علامت + کنار آن کلیک کنید.



شکل ۴۱-۵ - ویژگی موقعیت

در جلوی ویژگی Location، دو عدد ° و ° نوشته شده است. این اعداد فاصله کنترل را از سمت چپ فرم و از بالای فرم بر حسب پیکسل^۱ مشخص می کنند. به عبارت دیگر موقعیت یک کنترل نسبت به گوشه بالا سمت چپ^۲ فرم سنجیده می شود. بر خلاف آنچه که در ریاضیات معمول است که موقعیت یک نقطه را نسبت به مرکز صفحه (مبدأ) در نظر می گیرند، در کامپیوتر مبدأ سنجش موقعیت، گوشه بالا سمت چپ صفحه است.

برای تنظیم محل قرارگیری کنترل می توانید از ماوس استفاده کنید و یا اعداد مناسبی را به

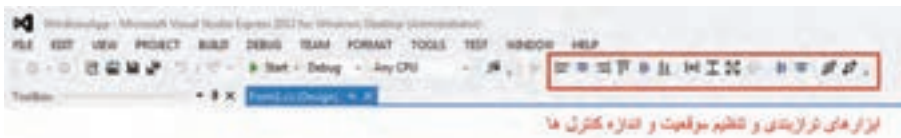


شکل ۵-۴۲- تنظیم موقعیت افقی با استفاده از منوی Format

عنوان فاصله از چپ و بالا مشخص کنید. در اینجا با استفاده از ماوس، کنترل برچسب را گرفته و در نقطه دلخواه فرم قرار دهید به طوری که از دو طرف فرم به یک اندازه باشد. همچنین می توانید مانند شکل ۵-۴۲، از منوی Format گزینه Center in Form استفاده کنید تا با دقت بیشتری کنترل را در وسط فرم قرار دهید.

نکته

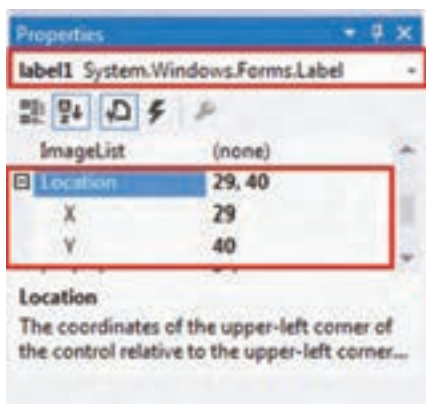
هنگامی که بیش از یک کنترل در روی فرم قرار داشته باشد، اگر آنها را با هم انتخاب کنید، ابزارهای تنظیم موقعیت آنها نسبت به یکدیگر که در روی نوار ابزار و همچنین در منوی Format وجود دارد فعال شده، به سادگی و با دقت و سرعت عمل بالا می توانید آنها را تنظیم کنید.



شکل ۵-۴۳

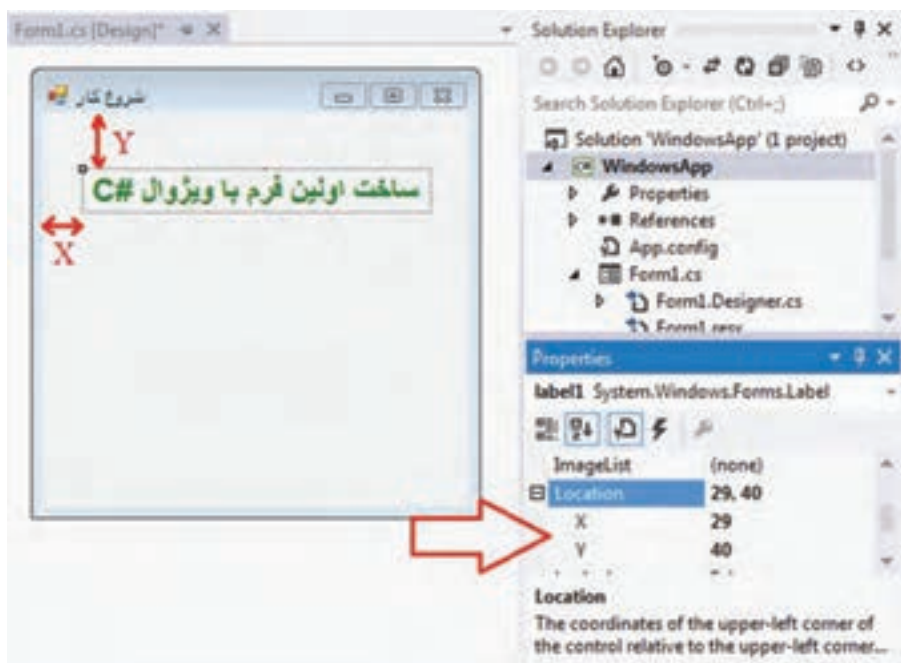
۱- Pixel

۲- Upper-left corner



شکل ۵-۴۴- مقادیر ویژگی موقعیت کنترل برچسب

بعد از جا به جایی کنترل برچسب و قرار گرفتن در یک موقعیت مناسب، به پنجره Properties رفته و مقدار اعداد در ویژگی Location را مشاهده کنید. همان طور که در شکل ۵-۴۴ مشاهده می کنید، در اثر جا به جایی کنترل، مقدار ° و ° به اعداد دیگری مانند ۲۹, ۴۰ تغییر کرده است. ارتباط این دو عدد با محل قرار گیری برچسب نسبت به فرم در شکل ۵-۴۵ نشان داده شده است. فاصله افقی^۱ نسبت به لبه سمت چپ فرم با مقدار X و فاصله عمودی^۲ نسبت به لبه بالایی فرم با مقدار Y مشخص می گردد.



شکل ۵-۴۵- ارتباط مقادیر ویژگی موقعیت با لبه های فرم

۱_ Horizontally

۲_ Vertically

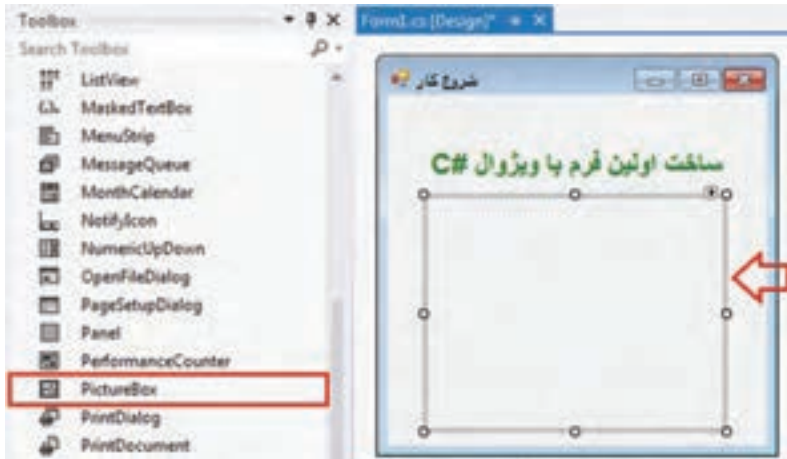
قبل از رفتن به مرحله بعدی بهتر است یک بار ویژگی‌ها و ویژگی‌های کنترل برچسب را که تنظیم کرده ایم در جدول ۵-۱ مشاهده کنید. رنگ نوشته و نوع فونت ممکن است با آنچه شما در انجام این مثال انجام می‌دهید، متفاوت باشد که اهمیتی ندارد.

جدول ۵-۱- مقادیر ویژگی‌های کنترل برچسب

کنترل Label	
ویژگی	مقدار
Name	label1
Text	ساخت اولین فرم با ویزوال C#
RightToLeft	Yes
Font	Arial Size : 16 Bold
ForeColor	Green
Location	Center in Form Horizontally

در سطر اول جدول ۵-۱، نام کنترل نوشته شده است و در ستون اول نام ویژگی‌های کنترل و در ستون دوم مقدار هر ویژگی بیان شده است. با توجه به اینکه جدول ۵-۱ به طور واضح و گویا ویژگی‌های کنترل برچسب را نشان می‌دهد، از چنین جداولی در انجام پروژه‌های بعدی و یا حتی در امتحانات عملی، برای مشخص کردن ویژگی‌های هر کنترل که باید تنظیم شود، استفاده خواهیم کرد و دیگر لازم به ارائه توضیحات نیست.

۶- اضافه کردن عکس بر روی فرم: در این مرحله باید عکسی را به فرم اضافه نماییم. برای نمایش یک عکس می‌توانید از کنترل PictureBox، استفاده کنید که یک کنترل جالب و کاربردی است و آن را کنترل جعبه تصویر می‌نامیم. برای قراردادن این کنترل بر روی فرم، به جعبه ابزار مراجعه کرده، آن را پیدا کرده و به فرم اضافه کنید. شکل ۵-۴۶ را مشاهده کنید.



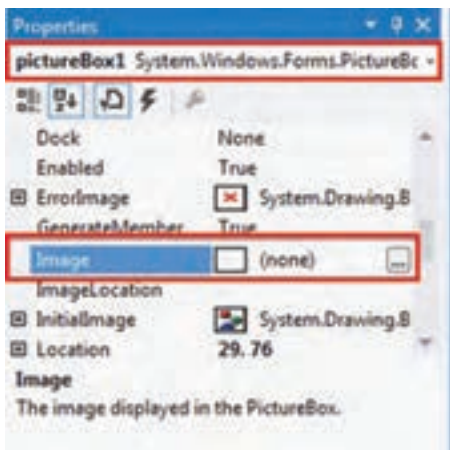
شکل ۴۶-۵ - اضافه کردن کنترل جعبه تصویر به فرم

به محض قرار گیری این کنترل بر روی فرم، پنجره Properties به طور خودکار، ویژگی‌ها این کنترل را نشان می‌دهد. نام این شیء pictureBox1 است که به وسیلهٔ VS انتخاب می‌شود و از کلاس PictureBox ساخته شده است.

برای تعیین تصویری که می‌خواهیم در جعبه تصویر نمایش داده شود، از ویژگی Image این کنترل استفاده می‌کنیم. برای مقداردهی ویژگی Image دو روش وجود دارد:

۱- استفاده از پنجره Properties، انتخاب شیء PictureBox و سپس کلیک روی علامت...

(شکل ۴۷-۵).



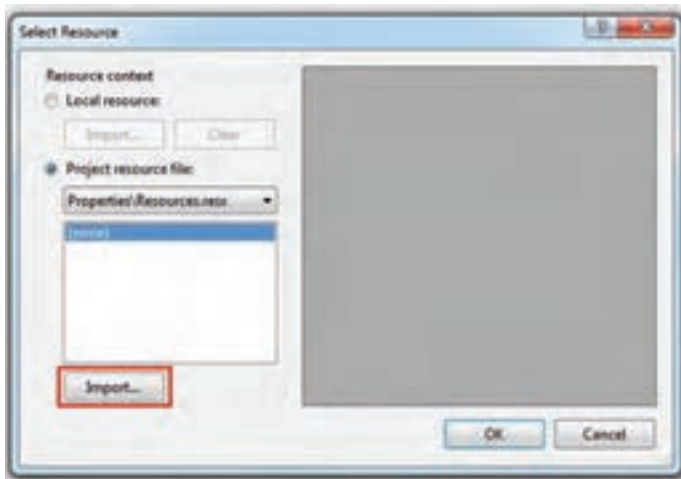
شکل ۴۷-۵ - ویژگی Image برای تعیین تصویر

۲- کلیک روی مثلث کوچک سیاه رنگ در کادر اطراف کنترل جعبه تصویر و استفاده از گزینه Choose Image در کادر اطراف کنترل جعبه تصویر، یک مثلث کوچک سیاه رنگ دیده می‌شود. با کلیک بر آن، منویی مانند شکل ۴۸-۵ ظاهر می‌گردد که در آن چند ویژگی کنترل نشان داده می‌شود.



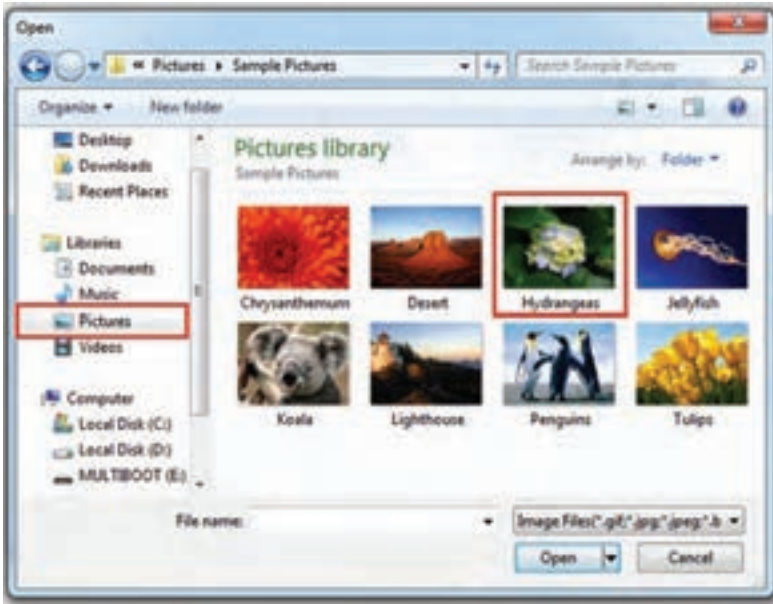
شکل ۵-۴۸ - کلیک بر روی مثلث و ظاهر شدن منو

به وسیله گزینه Choose Image در شکل ۵-۴۸ نیز علاوه بر پنجره Properties، می توانید ویژگی Image را تعیین کنید. در هر حال با انتخاب گزینه Choose Image، پنجره‌ای به نام انتخاب منبع (Select Resource)، شکل ۵-۴۹ باز می‌شود که در آن باید فایل مورد نظر برای اضافه شدن به پروژه را مشخص کنید. اگر در شکل ۵-۴۹ بر روی کلید Import کلیک کنید پنجره انتخاب فایل باز می‌شود تا یک کپی از فایل مورد نظر شما به لیست فایل‌های پروژه اضافه شود.



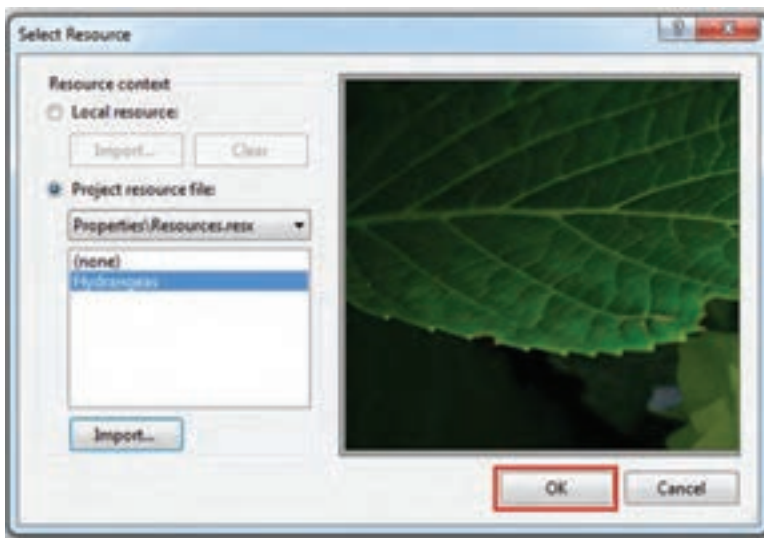
شکل ۵-۴۹ - انتخاب فایل تصویری جهت اضافه شدن به پروژه

پس از اینکه پنجره مربوط به تعیین نام فایل ظاهر شد وارد پوشه Pictures شوید و یک فایل تصویری انتخاب کنید. در این مثال تصویر گل ادریسی انتخاب شده است.



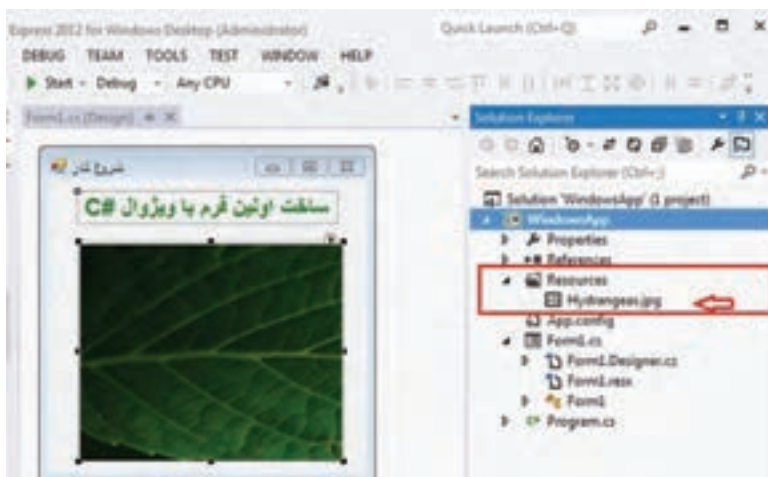
شکل ۵-۵۰ - انتخاب فایل تصویری جهت نمایش در کنترل جعبه تصویر

پس از انتخاب فایل مورد نظر، به پنجره انتخاب منابع باز می‌گردید. با کلیک روی دکمه OK عملیات انجام شده را تأیید کنید (شکل ۵۱-۵).



شکل ۵-۵۱ - پنجره انتخاب منبع پس از انتخاب فایل تصویری

پس از انتخاب فایل تصویری و کلیک بر روی دکمه OK ، تصویر انتخاب شده در جعبه تصویر قرار می‌گیرد. اگر مانند این مثال، اندازه عکس، بزرگ‌تر از اندازه جعبه تصویر باشد، فقط بخشی از آن مانند شکل ۵-۵۲ دیده خواهد شد. نگران نباشید این مشکل را برطرف می‌کنیم!



شکل ۵-۵۲- اضافه شدن فایل تصویری در فولدر Resources

اگر به پنجره Solution Explorer در شکل ۵-۵۲، دقت کنید فایل تصویر انتخاب شده در پوشه Resources دیده می‌شود. اگر از طریق My Computer به مسیری بروید که در ابتدا، برای پروژه تعیین کرده‌اید، پوشه‌ای به نام Resources را پیدا خواهید کرد که در داخل آن یک کپی از فایل تصویر انتخاب شده قرار دارد. به همین دلیل به این پوشه، پوشه منابع گفته می‌شود.

برای برطرف کردن مشکل عدم تطبیق اندازه عکس و اندازه جعبه تصویر چه کنیم؟ شاید راه حل‌های مختلفی برای برطرف کردن اشکال فوق، از جمله کاهش اندازه عکس به وسیله برنامه‌های گرافیکی نظیر Paint و Photoshop و یا بزرگ کردن اندازه جعبه تصویر به طوری که بتواند تمام عکس را در خود جای دهد، به نظر برسد. اما بهترین راه حل این است که از ویژگی SizeMode کنترل جعبه تصویر استفاده کنیم و مقدار آن را مانند شکل ۵-۵۳ برابر StretchImage قرار دهیم.



شکل ۵-۵۳- تغییر خاصیت SizeMode به StretchImage

حال برنامه را اجرا می‌کنیم، خروجی آن مانند شکل ۵-۵۴ است.



شکل ۵-۵۴- خروجی اولین برنامه نوشته شده در محیط Visual C#

تبریک می‌گوییم که توانستید اولین برنامه ویندوزی با استفاده از VS را بنویسید. در این مثال یک خط برنامه نوشتید و همه دستورات این برنامه به وسیله VS به طور خودکار انجام شد. بر روی فایل Form1.Designer.cs دوبار کلیک کنید و مستقیم به سراغ متد InitializeComponent() بروید و روی علامت + کلیک کنید تا محتوای آن را ببینید. توجه داشته باشید که محتویات این فایل، مخصوص این متد را نباید بدون اطمینان و آگاهی تغییر دهید و یا دستکاری کنید. شکل ۵-۵۵ را مشاهده کنید. دستورات مربوط به برجسب و جعبه تصویر مشخص شده است. دستورات قسمت پایین در شکل ۵-۵۵ مربوط به چیست؟

```
/// Required method for Designer support. do not modify
```

```
/// the contents of this method with the code editor. محتویات این متد را با ادیتور تغییر ندهید
```

```
/// </summary>
```

```
private void InitializeComponent ()
```

```
{
```

```
    this.label1 = new System.Windows.Forms.Label ();
```

```
    this.pictureBox1 = new System.Windows.Forms.PictureBox ();
```

```
    ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit ();
```

```
    this.SuspendLayout ();
```

```
    // label1
```

```
    //
```

```
    this.label1.AutoSize = true;
```

```
    this.label1.Font = new System.Drawing.Font("Arial", 15.75F, System.Drawing.
```

```
    this.label1.ForeColor = System.Drawing.Color.Green;
```

```
    this.label1.Location = new System.Drawing.Point(12,13);
```

```
    this.label1.Name = "label1";
```

```
    this.label1.RightToLeft = System.Windows.Forms.RightToLeft.Yes;
```

```
    this.label1.Size = new System.Drawing.Size(227,24);
```

```
    this.label1.TabIndex = 0;
```

```
    this.label1.Text = "#C ساخت اولین فرم با ویژوال";
```

```
    //
```

```
    // pictureBox1
```

```
    //
```

```
    this.pictureBox1.Image = global::WindowsApp.Properties.Resources.Hydrangeas
```

```
    this.pictureBox1.Location = new System.Drawing.Point(12, 57);
```

```
    this.pictureBox1.Name = "pictureBox1";
```

```
    this.pictureBox1.Size = new System.Drawing.Size(233, 193);
```

```
    this.pictureBox1.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;
```

```
    this.pictureBox1.TabIndex = 1;
```

```
    this.pictureBox1.TabStop = false;
```

```
    //
```

```
    // Form1
```

```
    //
```

مقداردهی
ویژگی‌های
برجسب

مقداردهی
ویژگی‌های
جعبه تصویر


```

this. AutoScaleDimensions = new System. Drawing. SizeF (6F, 13F);
this. AutoScaleMode = System. Windows. Forms. AutoScaleMode. Font;
this. ClientSize = new System. Drawing. Size (284, 262);
this. Controls. Add (this. pictureBox1);
this. Controls. Add (this. label1);
this. Name = "Form1" ;
this. Text = «شروع کار» ;
((System. ComponentMode 1. InitializeComponent) (this. pictureBox1)). EndInit ();
this. ResumeLayout (false);
this. PerformLayout ();

```

شکل ۵-۵۵- محتویات متد InitializeComponent() در فایل Form1.Designer.cs

برای مطالعه

کنترل‌ها و ابزارهای شخص ثالث برای .NET.

علاوه بر کنترل‌هایی که در جعبه ابزار VS وجود دارد، کنترل‌های دیگر و ابزارهای پیشرفته‌ای نیز توسط شرکت‌های نرم‌افزاری که با شرکت مایکروسافت همکاری دارند، ساخته شده‌اند. به این نوع کنترل‌ها که قابلیت اضافه شدن به فرم‌های ویندوزی را دارند و توسط شرکت دیگری ساخته می‌شود، کنترل‌های شخص ثالث نامیده می‌شوند. بعضی از این ابزارها و کنترل‌ها در محیط ویژوال استودیو اضافه می‌شوند.



شرکت DevExpress از جمله شرکت‌هایی است که مجموعه کنترل‌ها و ابزارهای واسطه کاربری را برای تولیدکنندگان برنامه‌های کاربردی، از جمله برنامه‌نویسانی که از

ویژوال استودیو استفاده می‌کنند، فراهم و به صورت تجاری عرضه کرده است. با استفاده از این کنترل‌ها، واسطه‌های کاربری زیباتری، بدون کدنویسی می‌توان ایجاد کرد.

یکی از ابزارهایی که این شرکت به رایگان عرضه می‌نماید، CodeRush Express است

که می‌توانید آن را از آدرس زیر دانلود نمایید. با نصب این ابزار در محیط VS، امکانات بیشتری در هنگام برنامه‌نویسی فراهم می‌شود. مثلاً منوی Refactoring با گزینه‌های متنوعی را در اختیار خواهید داشت.

<https://www.devexpress.com/Products/CodeRush/>

کار در کارگاه ۳: تنظیم اندازه کنترل

تنظیم خودکار اندازه کنترل‌ها در هنگام اجرای برنامه

۱- پروژه WindowsApp که در این

فصل ساختید را اجرا کنید.



شکل ۵-۵۶ - تغییر اندازه پنجره برنامه در حال اجرا

۲- همان‌طور که در شکل ۵-۵۶ مشاهده می‌کنید، با استفاده از ماوس گوشه سمت راست پایین

پنجره برنامه را گرفته و به سمت راست بکشید تا پنجره برنامه بزرگ‌تر شود.

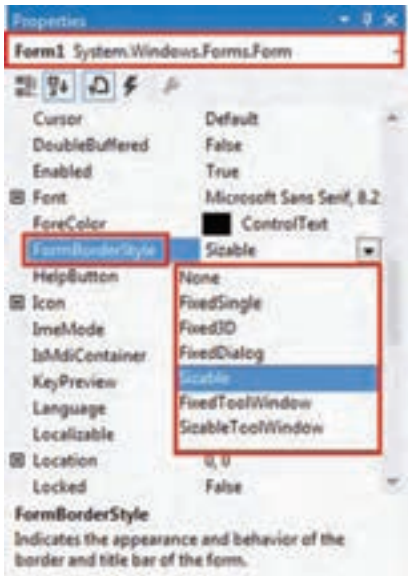


شکل ۵-۵۷

همان طور که در شکل ۵-۵۷ مشاهده می کنید، در اثر تغییر اندازه پنجره، فضای خالی در سمت راست ایجاد می شود. اما کاربران انتظار دارند با بزرگ شدن پنجره، اندازه تصویر نیز افزایش یابد و صفحه پر شود. ولی در برنامه بالا چنین نیست و توازن صفحه به هم می خورد.

برای رفع اشکال بالا، دو راه حل داریم:

الف) کاربر اجازه تغییر اندازه پنجره برنامه را نداشته باشد. در واقع صورت مسئله را پاک می کنیم



شکل ۵-۵۸- ویژگی FormBorderStyle فرم

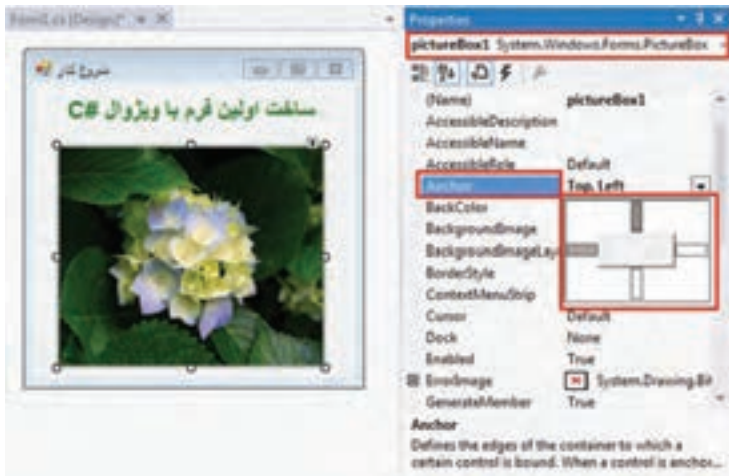
و به کاربر اجازه نمی دهیم که اندازه فرم را تغییر دهد تا چنین مشکلی به وجود آید. فرم ها دارای ویژگی به نام FormBorderStyle هستند که به وسیله آن می توانیم سبک یا شکل ظاهری کادر دور فرم را تعیین کنیم. در بعضی از سبک ها از امکان تغییر اندازه پنجره در حین اجرای برنامه جلوگیری می شود. مقدار پیش فرض این ویژگی Sizable است، یعنی فرم قابل تغییر اندازه است. برای انتخاب سبک های دیگر در حالی که فرم انتخاب شده است، در پنجره Properties به سراغ ویژگی FormBorderStyle رفته (شکل ۵-۵۸) و مقدار دیگری مانند FixedSingle که به معنای اندازه ثابت فرم و با کادر تک خطی می باشد را انتخاب کنید.

نکته

می توانید منوی کنترل پنجره و آیکن های بیشینه، کمینه و خروج را پنهان کنید. برای این منظور ویژگی ControlBox مربوط به فرم را برابر false کنید. البته در این صورت کاربر، مجبور است برای پایان دادن به اجرای برنامه از ترکیب کلیدی Alt + F4 استفاده کند.

ب) در روش دوم محدودیتی برای کاربر ایجاد نکرده و مانع تغییر اندازه فرم نمی شویم. بنابراین باید اندازه جعبه تصویر در هنگام تغییر اندازه پنجره، به طور خودکار افزایش یابد. و به عبارت جالب تر، باید فاصله جعبه تصویر از چهار لبه همواره ثابت بماند. خیلی نگران نباشید در VS برای انجام این

کار نیازی به برنامه نویسی خود شما نیست و فقط کافی است در محیط طراحی، از ویژگی Anchor کنترل جعبه تصویر برای این منظور استفاده کنید. این ویژگی‌ها می‌تواند مقادیر Left, Right, Top, Bottom و یا ترکیبی از آنها را بپذیرد. هر یک از مقادیر باعث می‌شوند که فاصله کنترل از سمت مربوطه ثابت بماند. در حالت پیش فرض مقدار ویژگی Anchor کنترل جعبه تصویر برابر Left, Top است، بنابراین فاصله کنترل از دو طرف چپ و بالای فرم، ثابت باقی می‌ماند. شکل ۵-۵۹ را مشاهده کنید.



شکل ۵-۵۹- ویژگی Anchor



شکل ۵-۶۰- تغییر اندازه پنجره برنامه در حال اجرا پس از تنظیم ویژگی Anchor کنترل PictureBox

اگر در ویژگی Anchor مربوط به جعبه تصویر هر چهار مقدار را قرار دهید (از ماوس کمک بگیرید و روی دو کادر خط چین راست و پایین در شکل ۵-۵۹ کلیک کنید تا رنگ آنها خاکستری شود)، فاصله آن از چهار طرف ثابت می‌ماند. به عبارت دیگر اندازه آن هماهنگ با افزایش اندازه پنجره تغییر می‌کند. شکل ۵-۶۰ پنجره برنامه را پس از افزایش توسط کاربر نشان می‌دهد.

۳- در شکل فوق مشاهده می‌کنید که برچسب در وسط پنجره قرار ندارد. برای اینکه موقعیت

برچسب نیز بتواند با تغییر اندازه پنجره تغییر کند و همواره در وسط باقی بماند لازم است فقط فاصله

آن تا بالای پنجره ثابت باشد. البته توجه داشته باشید برچسب باید از ابتدا در وسط فرم باشد. بنابراین برچسب را انتخاب کرده و ویژگی Anchor آن را برابر Top قرار دهید تا به طور خودکار موقعیت آن همواره در وسط پنجره قرار گیرد.

۴- برنامه را اجرا کنید و اندازه پنجره را تغییر دهید. آیا اندازه تصویر و موقعیت برچسب نیز مطابق با آن به طور خودکار تغییر می‌کند و کار رضایت بخش است؟

۵- در خیلی از برنامه‌های کاربردی نظیر Word و Excel که در سال قبل آشنا شدید، برای وارد کردن اطلاعات به زبان فارسی و یا نمایش صحیح اطلاعات فارسی، جهت صفحه را به حالت راست به چپ^۱ تغییر می‌دادید. در فرم‌ها و کنترل‌هایی که برای نمایش و یا دریافت اطلاعات فارسی به کار می‌روند نیز بهتر است ظاهر و ترتیب قرارگیری اجزای فرم و همچنین ترازبندی از راست به چپ تنظیم شوند. مثلاً در شکل ۵-۶۱ خط عنوان فرم به صورت راست به چپ تنظیم شده است.



شکل ۵-۶۱ - خط عنوان از راست به چپ

همان‌طور که در شکل مشاهده می‌کنید، خط عنوان فرم تغییر کرده است و حالت آینه‌ای نسبت به فرم‌های معمولی دارد. در این فرم، آیکن‌های بستن پنجره، بیشینه و کمینه در جهت سمت چپ قرار گرفته‌اند.

در فرم‌های ویندوزی خاصیت RightToLeft و خاصیت RightToLeftLayout برای پشتیبانی از زبان‌های راست به چپ، پیش‌بینی شده است. برای ایجاد چنین حالتی، باید مقدار هر دو خاصیت را به true تغییر دهید.

۱ - Right to Left

الف) درستی یا نادرستی هر عبارت را تعیین کنید.

- ۱- برنامه‌های کنسولی کاربرد زیادی در سیستم عامل ویندوز دارند.
 - ۲- در برنامه‌های ویندوزی، واسط کاربری به صورت گرافیکی است.
 - ۳- برای نمایش یک متن روی فرم از کنترل Label استفاده می‌کنیم.
 - ۴- عنوان یک فرم از طریق ویژگی Text مقداردهی می‌شود.
 - ۵- می‌توان منوی کنترل پنجره و آیکن‌های بیشینه، کمینه و خروج را پنهان کرد.
- ب) در سؤالات چند گزینه‌ای زیر، پاسخ صحیح را انتخاب نمایید.
- ۶- کدام ویژگی یک برجسب مربوط به متن داخل آن است؟

الف) RightToLeft (ب) Text

ج) Enabled (د) TextAlign

۷- برای تغییر ویژگی‌های یک کنترل از پنجره استفاده می‌کنیم.

الف) Solution Explorer (ب) Properties

ج) Toolbox (د) Class View

۸- برای دیدن پروژه و فایل‌های آن از پنجره استفاده می‌شود

الف) Solution Explorer (ب) Properties

ج) Toolbox (د) Class View

۹- اگر بخواهیم در محیط طراحی، یک کنترل را به فرم اضافه کنیم از پنجره آن را

به داخل فرم می‌آوریم.

الف) Solution Explorer (ب) Properties

ج) Toolbox (د) Class View

۱۰- ویژگی SizeMode برای تنظیم چه چیزی از کنترل PictureBox به کار برده می‌شود؟

الف) مکان تصویر (ب) تصویر درون کنترل

ج) نوع تعیین اندازه تصویر (د) اندازه کنترل

۱۱- برای ثابت نگه داشتن فاصله از طرفین برای یک کنترل از کدام ویژگی استفاده می‌نماییم؟

الف) Dock (ب) Text

- Location (ج) Anchor (د)
- ۱۲- اگر ویژگی FormBorderStyle برابر با Sizable شود فرم..... خواهد بود.
- الف) با اندازه ثابت ب) یک کادر محاوره‌ای
- ج) بدون دکمه‌های پیشینه و کمینه د) با اندازه متغیر
- ج) جاهای خالی را با عبارات مناسب پر کنید.
- ۱۳- اطلاعی که از طرف سیستم عامل به برنامه داده می‌شود..... نامیده می‌شود.
- ۱۴- ویژگی RightToLeft برای زبان‌های..... پیش‌بینی شده است
- ۱۵- در کنترل PictureBox، ویژگی که تصویر را در بر می‌گیرد..... است.

تمرینات برنامه نویسی فصل پنجم

۱- فرم زیر را برای نمایش اطلاعات شخصی خود (نام ، نام خانوادگی، نام پدر، تاریخ تولد و عکس) طراحی کنید.



نام	علی
نام خانوادگی	شوی
نام پدر	محمدرضا
تاریخ تولد	1377/1/10

۲- فرم تمرین ۱ برای نمایش اطلاعات به زبان فارسی که یک زبان RTL است، مناسب نیست. با تغییر ویژگی‌های فرم و برچسب‌ها، فرم را برای زبان RTL مناسب کنید.

متن زیر از MSDN با موضوع برنامه‌های ویندوزی برداشت شده است. آن را با کمک هم کلاسی خود ترجمه کنید و به کلاس ارائه دهید.

Windows Forms provide your project with components, such as dialog boxes, menus, buttons, and many other controls, that make up a standard Windows application user interface (UI). Fundamentally, these controls are just classes from the .NET Framework class library. The Designer view in Visual C# Express Edition enables you to drag the controls onto your application's main form and adjust their size and position. As you do this, the IDE automatically adds the source code to create an instance of the appropriate class and initialize it.

واژگان و اصطلاحات انگلیسی فصل پنجم

ردیف	واژه انگلیسی	معنی به فارسی
۱	Action	
۲	Alphabetical view	
۳	Button	
۴	Categorized view	
۵	container	
۶	Context menu	
۷	Event	
۸	Form Designer	
۹	Graphical User Interface	
۱۰	Horizontally	
۱۱	Message	
۱۲	Notification	
۱۳	Object	
۱۴	Pixel	
۱۵	Pushpin	
۱۶	Right to Left	
۱۷	Right to left languages	
۱۸	Shortcut	
۱۹	Timer Control	
۲۰	Tool Bar	
۲۱	Tool Box	
۲۲	Upper-left corner	
۲۳	Vertically	
۲۴	Wait state	

ایجاد برنامه‌های ویندوزی با واکنش نسبت به رویدادها

در فصل ۵ با برنامه‌های ویندوزی و کنترل‌های فرم، برجسب و جعبه تصویر آشنا شدید. در محیط VS برنامه‌های ویندوزی ایجاد کرده، فرم ساده طراحی کردید و بدون نوشتن حتی یک خط برنامه، آن را اجرا نمودید؛ در این فصل علاوه بر آشنایی با کنترل‌های جدید، برای رویدادها و عملیاتی که کاربر انجام می‌دهد برنامه نوشته می‌شود.

پس از پایان این فصل انتظار می‌رود که فراگیر بتواند:

- ۱- در برنامه‌های ویندوزی از کنترل دکمه استفاده نماید و ویژگی‌های مورد نیاز را تنظیم نماید.
- ۲- مفهوم رویداد را توضیح دهد و در رویداد کلیک ماوس، برنامه بنویسد.
- ۳- برای واکنش به رویدادها Event Handler بنویسد.
- ۴- به برنامه‌های خود کنترل جعبه متن اضافه کند و ویژگی‌های آن را تعیین کند.
- ۵- با استفاده از کنترل زمان سنج و تنظیم ویژگی‌های آن، برنامه‌های کاربردی بنویسد.
- ۶- با کنترل‌های Dialogs برای تغییر رنگ و قلم و تنظیمات پرونده، برنامه بنویسد.

کار در کارگاه ۱: آشنایی با کنترل دکمه

مثال ۱-۶: برنامه ویندوزی ساده با واکنش به رویداد کلیک

۱- یک پروژه جدید ویندوزی بسازید.

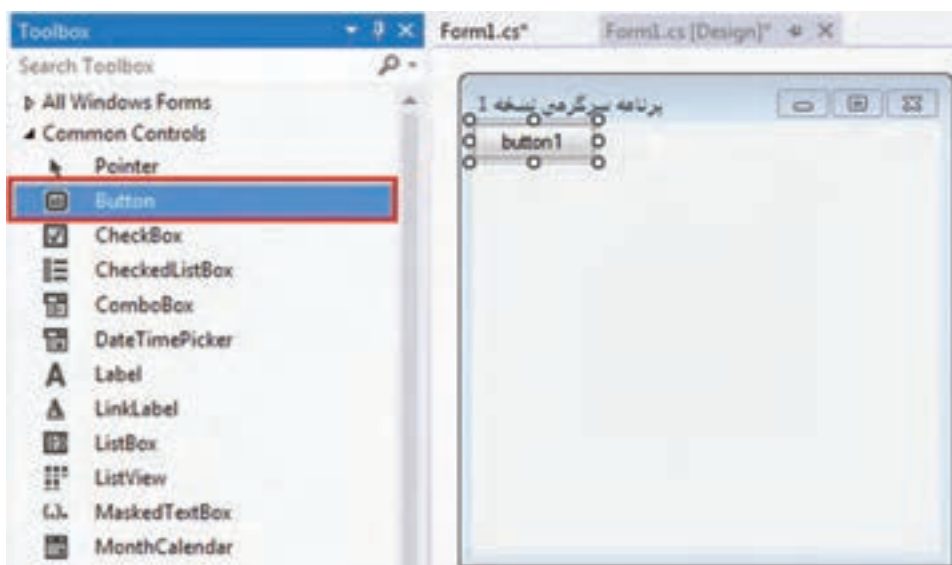
۲- ویژگی‌های فرم را مانند جدول ۱-۶ تنظیم کنید.

جدول ۱-۶- ویژگی‌های فرم

Form	
ویژگی	مقدار
Name	form1
Text	برنامه سرگرمی نسخه ۱
RightToLeft	Yes

۳- به جعبه ابزار مراجعه کرده و کنترل Button را پیدا کرده و آن را روی فرم قرار دهید. شکل

۱-۶ را مشاهده کنید.



شکل ۱-۶- قرار دادن کنترل Button روی فرم

۴- ویژگی‌های دکمه را مطابق جدول زیر تنظیم کنید.

جدول ۶-۲- ویژگی‌های Button

Button	
ویژگی	مقدار
Name	button1
Text	لبخند بزنید
RightToLeft	Yes
Font	Arial Size : 16
AutoSize	true
Location	Center in Form Horizontally Center in Form Vertically

۵- اکنون برنامه را اجرا کنید باید فرم ایجاد شده مانند شکل ۶-۲ باشد.



شکل ۶-۲

۶- با اجرای برنامه و کلیک کردن بر روی دکمه ، هیچ اتفاقی نمی افتد. نه تنها لبخندی در کار نیست شاید کاربر کمی هم دلگیر شود که چرا برنامه، کار خاصی انجام نمی دهد. در برنامه بعدی با روش ایجاد عکس العمل و واکنش نسبت به رویداد کلیک آشنا می شوید.

۷- گاهی اوقات علامت ها و نمادها می توانند خیلی بهتر از نوشته ها پیام خود را برسانند. مثلاً یک کنترل تلویزیون را در نظر بگیرید، در روی بعضی از دکمه های آن، علامت هایی حک شده است که به سرعت عملکرد کلید را نشان می دهد. در این مرحله می خواهیم به جای عبارت «لبخند بزن» یک نماد مثلاً 😊 را قرار دهیم.

۸- دو ویژگی Font و Text دکمه را که در جدول ۳-۶ با رنگ قرمز نشان داده شده است تغییر دهید. توجه داشته باشید که در ویژگی Text، حرف J بزرگ است.

جدول ۳-۶- ویژگی های Button

Button	
ویژگی	مقدار
Name	button1
Text	J
RightToLeft	No
Font	Wingdings Size : 28
AutoSize	True
Location	Center in Form Horizontally Center in Form Vertically

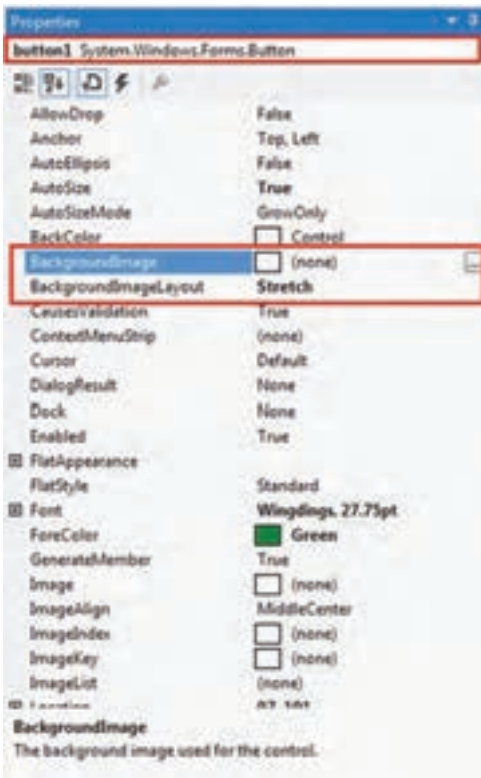
۹- اکنون برنامه را اجرا کنید. نتیجه کار باید شبیه شکل ۳-۶ باشد

سؤال: چگونه می‌توانید رنگ نماد
لیخند، را مانند شکل ۳-۶ به رنگ سبز درآورید؟



شکل ۳-۶

۱۰- ویژگی Text دکمه را به حرف K (بزرگ) تغییر دهید و برنامه را اجرا کنید. اگر مایل هستید یک بار هم با حرف L (بزرگ) امتحان کنید. اگر به استفاده از این نمادها علاقمند



شکل ۴-۶ ویژگی‌های کنترل Button

هستید، وارد برنامه Word شوید و فونت را Wingdings قرار داده و با زدن کلیدهای صفحه کلید، نمادهای مختلفی را شناسایی کرده و از آنها در برنامه خود استفاده نمایید.

۱۱- علاوه بر نوشته یا نماد، می‌توانید برای روشن شدن کاربرد یک دکمه، تصویری را نیز به عنوان زمینه دکمه با استفاده از ویژگی BackgroundImage معین کنید. اگر اندازه تصویر مورد نظر با اندازه دکمه متفاوت است، مقدار ویژگی BackgroundImageLayout را برابر با Stretch قرار داده تا کل تصویر در روی دکمه جای داده شود. توجه داشته باشید که اگر متنی را نیز در ویژگی Text نوشته باشید، متن روی عکس زمینه قابل مشاهده است. شکل ۴-۶ را مشاهده کنید.

۱۲- یکی دیگر از ویژگی‌هایی که در شکل ۴-۶ مشاهده می‌کنید، ویژگی Image است که برای انتخاب یک تصویر بر روی دکمه استفاده می‌شود. در صورت استفاده از چنین ویژگی، نوشته موجود در ویژگی Text دیده نمی‌شود چون تصویر روی آن را می‌پوشاند.

کار در کارگاه ۲: کار با دکمه

در این بخش خواهید دید که VS، در تولید برنامه‌هایی که نسبت به عملیات کاربر، واکنش نشان می‌دهد، بسیاری از دستورات و عملیات را در پشت صحنه انجام می‌دهد و برنامه نویس تنها با چند کلیک ماوس و نوشتن دستورات متد مربوطه، می‌تواند برنامه‌های واکنش دار را بسازد.

مثال ۲-۶: ایجاد یک فرم شامل یک دکمه و واکنش نسبت به رویداد کلیک ماوس
در این مثال می‌خواهیم با اجرای برنامه، فرمی مطابق با شکل ۵-۶ نشان داده شود که ظاهراً فقط شامل یک دکمه است. برنامه در وضعیت انتظار باقی می‌ماند تا کاربر، عملی را انجام دهد. هنگامی که کاربر بر روی دکمه «بگو سلام» کلیک کرد، پیامی مطابق با شکل ۶-۶ ظاهر می‌گردد.



شکل ۶-۶ - نمایش پیام به محض کلیک بر روی دکمه



شکل ۵-۶ - برنامه در حالت انتظار

مراحل اجرای این مثال را قدم به قدم به صورت زیر انجام می‌دهیم:

۱- ایجاد پروژه: وارد برنامه VS شوید و یک پروژه جدید از نوع Windows Forms Application با نام پروژه EventDemo و در مسیر مشخصی بسازید.

۲- تعیین ویژگی های فرم : ویژگی های فرم را به صورت جدول ۴-۶ تنظیم کنید.

جدول ۴-۶ - ویژگی های فرم

Form		
ویژگی	مقدار	
Name	Form1	
Text	واکنش به رویداد کلیک	
Size	Width	300
	Height	300

۳- اضافه کردن یک متن نمایشی بر روی فرم : یک کنترل برجسب بر روی فرم قرار دهید و ویژگی های آن را مانند جدول ۵-۶ تعیین کنید.

جدول ۵-۶ - ویژگی های برجسب

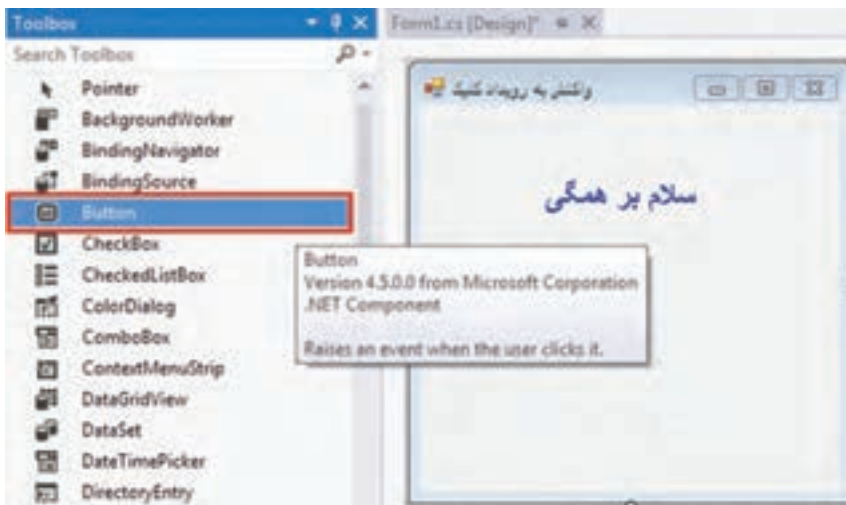
Label		
ویژگی	مقدار	
Name	Label1	
Text	سلام بر همگی	
RightToLeft	True	
Font	Arial Size : 16 Bold	
ForeColor	Blue	
Location	X	Center in Form Horizontally
	Y	50
Visible	false	

اگر برنامه را در این مرحله اجرا کنید، مشاهده خواهید کرد که فقط یک فرم خالی دیده می‌شود. شاید تعجب کنید که، با وجود اینکه در روی فرم، یک کنترل برچسب قرار دادید، اما در هنگام اجرای برنامه دیده نمی‌شود. فکر می‌کنید علت چیست؟

اگر به ردیف آخر جدول ۵-۶ دقت کنید، متوجه می‌شوید که یک ویژگی جدید به نام Visible استفاده شده است که مقدار آن را false قرار داده‌ایم. با توجه به معنی کلمه Visible (قابل مشاهده) و مقداری که برای این ویژگی تعیین کردیم، باید انتظار داشت که کنترل برچسب نشان داده نشود. هرگاه مقدار true در این ویژگی قرار دهیم کنترل برچسب نشان داده می‌شود.

حال طراحی برنامه را ادامه می‌دهیم و در این مرحله می‌خواهیم کاری کنیم تا کاربر با کلیک بر روی ماوس، برچسب را نمایان کند.

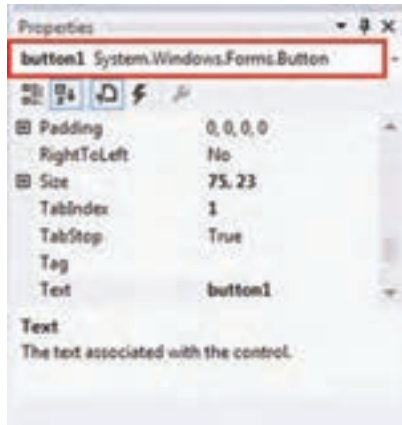
۴- اضافه کردن دکمه بر روی فرم : در این مرحله یک دکمه به فرم اضافه می‌کنیم برای این منظور به جعبه ابزار مراجعه و کنترل دکمه را مطابق شکل ۷-۶ پیدا کرده و به فرم اضافه کنید.



شکل ۷-۶- انتخاب کنترل دکمه از جعبه ابزار

به محض قرارگیری کنترل دکمه بر روی فرم، پنجره Properties به طور خودکار، ویژگی‌های این کنترل را نشان می‌دهد. نام شیء دکمه، button1 است که به وسیلهٔ VS انتخاب می‌شود. این شیء از کلاس Button ساخته شده است.

حال ویژگی‌های دکمه را مانند جدول ۶-۶ تنظیم کنید.



شکل ۸-۶ پنجره Properties برای نمایش ویژگی‌های دکمه

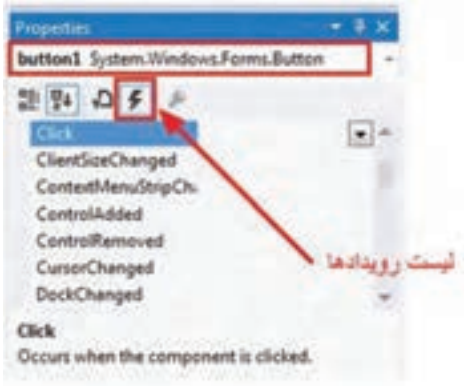
جدول ۶-۶- ویژگی‌های Button

دکمه Button		
ویژگی	مقدار	
Name	button1	
Text	بگو سلام	
Location	X	Center in Form Horizontally
	Y	200

حال برنامه را اجرا و روی دکمه «بگو سلام» کلیک کنید، مشاهده می‌شود که هیچ اتفاقی نمی‌افتد. طبیعی است که برنامه عکس‌العملی نسبت به رویداد کلیک نداشته باشد چون هیچ متدی برای رویداد کلیک بر روی دکمه ایجاد نکردیم و همچنین هیچ دستوری برای نمایش پیام و تغییر ویژگی Visible برچسب نوشته نشده است. حال از برنامه خارج شده و به محیط طراحی فرم باز می‌گردیم.

در پنجره Properties علاوه بر لیست ویژگی‌ها، می‌توانید لیست رویدادهای مربوط به دکمه را نیز مشاهده کنید. با کلیک بر روی آیکن رعد و برق، مانند شکل ۹-۶ لیست رویدادها مشاهده خواهد

شد. مقدار تمام رویدادها خالی است، یعنی هیچ واکنشی نسبت به رویدادها ثبت نشده است.



شکل ۹-۶ پنجره Properties برای نمایش رویدادهای دکمه

در ردیف اول لیست رویدادها، رویداد Click را مشاهده می‌کنید. این رویداد مربوط به کلیک ماوس است. در مرحله بعدی خواهید دید که چگونه واکنشی نسبت به این رویداد ثبت می‌کنیم.

۵- واکنش برنامه نسبت به رویداد کلیک ماوس: هنگامی که کاربر بر روی دکمه «بگو سلام»، کلیک کرد، باید پیام «سلام بر همگی» دیده شود. برای انجام این کار، باید دستوری اجرا گردد تا ویژگی Visible برچسب برابر true شود.

```
label1.Visible = true;
```

سؤال؟ این دستور کجا باید نوشته شود؟

برای نوشتن برنامه‌هایی که باید به عمل کاربر واکنش نشان دهند، لازم است دو مرحله زیر انجام شود:

۱- نوشتن یک متد (EH)

۲- ثبت کردن رویداد ماوس و برقرار کردن ارتباط رویداد و متد

خوشبختانه VS در انجام دو عمل فوق، شما را تنها نمی‌گذارد و بیشتر عملیات را خودش انجام می‌دهد فقط بایستی دستورات داخل متد EH را بنویسیم.

مرحله اول: نوشتن متد EH: برای اینکه بتوانید متد را بنویسید در حالی که در صفحه طراحی فرم هستید، روی دکمه «بگو سلام» دوبار کلیک کنید. در این صورت پنجره کد نویسی مانند شکل ۱۰-۶ باز می‌شود که در آن دستورات مربوط به خط عنوان متد به طور خودکار نوشته شده است.

```

namespace EventDemo
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
        }
    }
}

```

شکل ۱۰-۶- پنجره کد نویسی برای رویداد کلیک دکمه

نام متد، button1_Click است که به وسیلهٔ VS انتخاب شده است. فقط بایستی در داخل متد، دستورات مورد نظر را بنویسیم. بنابراین دستور را مانند برنامه ۱-۶ می‌نویسیم:

```

using System. Collections. Generic;
using System. ComponentMode 1;
using System. Drawing;
using System. Text;
using System. Windows. Forms;

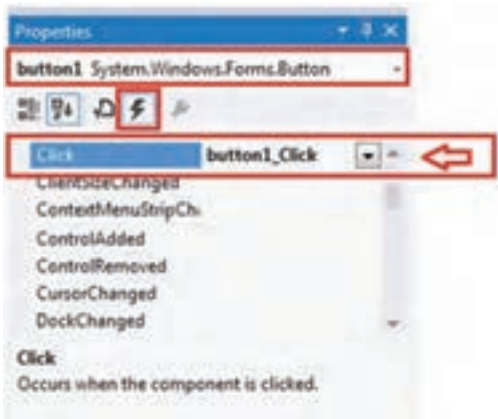
```

```

namespace EventDemo
{
    public partial class Form1: Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click (object sender, EventArgs e)
        {
            label1. Visible = true; ←
        }
    }
}

```



شکل ۱۱-۶ - تعیین متد EH مربوط به رویداد کلیک دکمه

مرحله دوم : ثبت متد EH

برای رویداد کلیک : باید متد EH را برای یک رویداد ثبت کنیم. خوشبختانه این کار را VS به طور خودکار انجام می دهد و شما هیچ کاری لازم نیست انجام دهید. در این برنامه، VS به طور خودکار متد `button1_Click()` را برای رویداد `Click` دکمه ثبت می کند. برای اینکه مطمئن شوید که این کار صورت گرفته، کافی است به صفحه طراحی فرم برگشته و

در پنجره Properties بر روی آیکون رعد و برق کلیک نمایید تا لیست رویدادها ظاهر گردد. در لیست به رویداد `Click` توجه کنید. در جلوی رویداد `Click`، نام متدی نوشته شده است که در آن متد، دستور ظاهر کردن قرار دارد. شکل ۱۱-۶ را مشاهده کنید.

آکنون برنامه را اجرا کنید و رویداد کلیک را آزمایش نمایید. از برنامه خارج شده و آن را مجدداً اجرا نمایید.

سؤال؟ چگونه از طریق پنجره Properties، می توانید واکنش نسبت به رویداد `Click` دکمه را از کار بیاندازید؟

تبریک می گوئیم که توانستید اولین برنامه ویندوزی با واکنش نسبت به کلیک ماوس را با همیاری VS بنویسید. در این مثال فقط یک خط دستور نوشتید و بقیه عملیات را با ماوس انجام دادید.

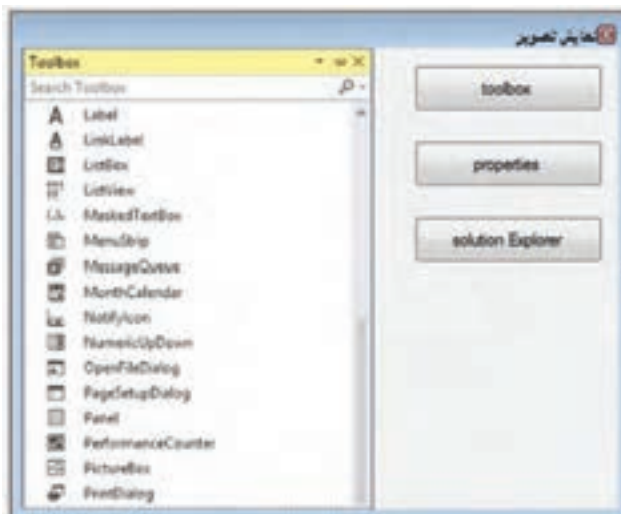
توسعه و بهبود برنامه : یک جعبه تصویری به فرم قبل اضافه کنید به طوری که بین پیام و دکمه مانند شکل ۱۱-۶ قرار گیرد. تصویر گل لاله^۱ را در آن قرار دهید و اگر لازم است موقعیت برچسب و دکمه را کمی تغییر دهید تا تصویر بهتر در روی فرم جای گیرد. با اجرای برنامه، یک فرم خالی نمایش داده شود (تصویر و پیام هر دو دیده نشوند) اما هنگامی که کاربر بر روی دکمه «بگو سلام» کلیک کرد هر دو نشان داده شوند.



شکل ۱۲-۶ - اضافه کردن یک جعبه تصویری

کار در کارگاه ۳: کار با جعبه تصویر

مثال ۳-۶: ایجاد فرمی با سه دکمه و یک جعبه تصویر برای نمایش سه تصویر در این مثال می‌خواهیم فرمی مانند شکل ۱۳-۶ با سه دکمه ایجاد کنیم و سه تصویر مختلف را در PictureBox نمایش دهیم به صورتی که با کلیک روی هر دکمه، تصویر متناظر با آن دکمه در PictureBox قرار گیرد. ما از تصویر پنجره‌های Toolbox، Properties و Solution محیط IDE ویژوال استودیو استفاده می‌کنیم.



شکل ۱۳-۶ - فرمی با سه دکمه و PictureBox برای نمایش تصویر

۱- ایجاد پروژه: وارد برنامه VS شوید و پروژه جدیدی از نوع Windows Forms Application با نام Picture Show در مسیر مشخص بسازید.

۲- تعیین ویژگی های فرم

جدول ۶-۷- ویژگی های فرم

Form		
ویژگی		مقدار
Name		Form1
Text		نمایش تصویر
FormBorderStyle		FixedToolWindow
RightToLeft		Yes
Size	Width	440
	Height	362

۳- تعیین ویژگی دکمه ها و جعبه تصویر

جدول ۶-۹- ویژگی های دکمه دوم

Button2		
ویژگی		مقدار
Name		toolboxBtn
Text		toolbox
Size	Width	134
	Height	33

جدول ۶-۸- ویژگی های دکمه اول

Button1		
ویژگی		مقدار
Name		propertiesBtn
Text		properties
Size	Width	134
	Height	33

جدول ۱۰-۶ ویژگی های دکمه سوم

Button3		مقدار	ویژگی
Name		pic	
Text		solution Explorer	
Size	Width	134	
	Height	33	

جدول ۱۱-۶ ویژگی های جعبه تصویر

PictureBox		مقدار	ویژگی
Name		pic	
Dock		Left	
SizeMode		StretchImage	
Size	Width	250	
	Height	328	

ویژگی Image کنترل PictureBox را در پنجره Properties پیدا کرده روی علامت ... کلیک کنید تا جعبه محاوره ای انتخاب منبع (SelectResource) باز شود با استفاده از دکمه Import سه تصویر را به لیست اضافه کنید و در پایان یکی از تصاویر را از لیست انتخاب کرده و روی دکمه Ok کلیک کنید.

تصاویر اضافه شده در پنجره Solution Explorer در پوشه Resources قرار می گیرد.

۴- ایجاد واکنش نسبت به رویداد کلیک ماوس : با کلیک هر یک از دکمه ها تصویر PictureBox باید تغییر کند بنابراین باید ویژگی Image آن را تغییر دهید. همانطور که قبلاً مشاهده کردید در هنگام انتخاب عکس برای کنترل pictureBox در جعبه محاوره ای انتخاب منبع، گزینه Project resource file را انتخاب کردید. به این ترتیب با وارد کردن یک یا چند تصویر، یکی را به عنوان تصویر مورد نظر خود انتخاب نمودید. در واقع با این کار، منابعی از جنس تصویر برای نرم افزار ایجاد کردید و از آن استفاده نمودید. این منابع در داخل نرم افزار شما جایگذاری می شوند. اگر بخواهید از منابع معرفی شده در نرم افزار خود در زمان اجرا (در کد) استفاده کنید باید از کلاس Resources که در فضای نام Properties است، به صورت زیر استفاده نمایید.

نام منبع . Resources . Properties

در اینجا منابع ما از جنس تصویر (image) هستند و برای این مثال، چون نام تصویر جعبه ابزار toolbox بوده است به همین نام در منابع موجود است و از آن می‌توانیم استفاده نماییم:

Properties.Resources.toolbox

در فضای طراحی فرم روی دکمه‌ها دابل کلیک کنید تا الگوی رویداد کلیک آنها نوشته شود، سپس کد را مانند زیر تکمیل نمایید.

```
private void propertiesBtn_Click(object sender, EventArgs e)
{
    pic.Image = Properties.Resources.properties;
}
```

```
private void solutionBtn_Click(object sender, EventArgs e)
{
    pic.Image = Properties.Resources.solution;
}
```

```
private void toolboxBtn_Click(object sender, EventArgs e)
{
    pic.Image = Properties.Resources.toolbox;
}
```

برنامه ۶-۲

توسعه و بهبود برنامه: در برنامه فوق یک کنترل برچسب اضافه کنید که نام تصویری را نشان دهد که در PictureBox قرار دارد؟

کار در کارگاه ۴: بزرگ کردن تصویر

مثال ۴-۶: در این مثال می‌خواهیم فرمی مانند شکل ۶-۱۴ با سه دکمه ایجاد کنیم و سه تصویر مختلف را روی دکمه‌ها قرار دهیم و با کلیک روی هر دکمه، تصویر آن دکمه در PictureBox قرار گیرد. در این مثال به جای متن، روی دکمه‌ها تصویر قرار می‌دهیم.



شکل ۱۴-۶- فرمی با سه دکمه و جعبه تصویر برای نمایش تصویر

- ۱- ایجاد پروژه: وارد برنامه VS شوید و پروژه جدیدی از نوع `WindowsFormsApplication` با نام `PictureZoom` در مسیر مشخص بسازید.
- ۲- تعیین ویژگی های فرم

جدول ۱۲-۶- ویژگی های فرم

Form		مقدار
ویژگی		
Name		Form1
Text		بزرگ کردن تصویر
RightToLeft		Yes
Size	Width	440
	Height	362

۳- تعیین ویژگی های دکمه ها و جعبه تصویر

جدول ۱۳-۶- ویژگی های دکمه اول

Button1		
ویژگی		مقدار
Name		pic1Btn
Text		
BackgroundImage		تصویر دلخواه ۱
BackgroundImageLayout		Stretch
Size	Width	126
	Height	90

جدول ۱۴-۶- ویژگی های دکمه دوم

Button2		
ویژگی		مقدار
Name		pic2Btn
Text		
BackgroundImage		تصویر دلخواه ۲
BackgroundImageLayout		Stretch
Size	Width	126
	Height	90

جدول ۱۵-۶- ویژگی های دکمه سوم

Button3		
ویژگی		مقدار
Name		pic3Btn
Text		
BackgroundImage		تصویر دلخواه ۳
BackgroundImageLayout		Stretch
Size	Width	126
	Height	90

جدول ۱۶-۶- ویژگی های جعبه تصویر

PictureBox		
ویژگی		مقدار
Name		piclarge
SizeMode		StretchImage
Size	Width	250
	Height	328

برای قرار دادن تصویر روی دکمه ها، در فضای طراحی روی دکمه کلیک کنید تا در پنجره Properties ویژگی های آن نمایش داده شود. ویژگی BackgroundImage دکمه را پیدا کرده و روی علامت ... کلیک کنید تا جعبه محاوره ای انتخاب منبع باز شود و سپس با استفاده از دکمه Import

سه تصویر را به لیست اضافه کنید. تصاویر اضافه شده در پنجره Solution Explorer در پوشه Resources قرار می‌گیرد. در پایان یکی از تصاویر اضافه شده به لیست را انتخاب کرده و روی دکمه Ok کلیک کنید.

برای اینکه تصویر روی دکمه‌ها و PictureBox کامل دیده شود باید ویژگی BackgroundImageLayout دکمه‌ها و PictureBox SizeMode کنترل PictureBox را مساوی Stretch قرار دهید.

مقدار ویژگی Text دکمه‌ها را پاک کنید تا روی تصاویر متنی نباشد و فقط تصویر زمینه دیده شود.

۴- ایجاد واکنش نسبت به رویداد کلیک ماوس: با کلیک هر یک از دکمه‌ها، تصویر دکمه باید در PictureBox نمایش داده شود. بنابراین باید ویژگی Image آن را مساوی BackgroundImage دکمه قرار دهیم.

درفضای طراحی فرم روی دکمه‌ها دابل کلیک کنید تا الگوی رویداد کلیک آنها نوشته شود، سپس کد را مانند زیر تکمیل نمایید.

```
private void pic1Btn_Click(object sender, EventArgs e)
{
    piclarge.Image = pic1Btn.BackgroundImage;
}

private void pic2Btn_Click(object sender, EventArgs e)
{
    piclarge.Image = pic2Btn.BackgroundImage;
}

private void pic3Btn_Click(object sender, EventArgs e)
{
    piclarge.Image = pic3Btn.BackgroundImage;
}
```

برنامه ۳-۶

مثال ۵-۶: حرکت دادن برجسب با کمک دکمه‌ها

در این مثال می‌خواهیم فرمی مانند شکل ۱۵-۶ طراحی کنیم که کاربر بتواند یک برجسب را در چهار جهت اصلی بالا، پایین، چپ و راست حرکت دهد و از جهت آخرین حرکت برجسب با اطلاع باشد.

الگوریتم و روش انجام کار: برای تعیین جهت حرکت از دکمه استفاده می‌کنیم. با کلیک روی هر دکمه فرمان، برجسب در جهت متناظر با آن دکمه فرمان، ۱۰ واحد جابه‌جا می‌شود و جهت حرکت نیز روی برجسب نمایش داده می‌شود. بنابراین به چهار دکمه فرمان و یک کنترل برجسب نیاز داریم.



شکل ۱۵-۶ - فرم جابه‌جایی برجسب

۱- ایجاد پروژه: وارد برنامه VS شوید و پروژه جدیدی از نوع `WindowsFormsApplication` به نام `MoveLabelProj` ایجاد نمایید.

۲- درج کنترل‌ها: روی فرم چهار کنترل دکمه و یک برجسب قرار دهید.

۳- تنظیم ویژگی‌های برجسب و فرم: ویژگی‌های فرم و برجسب را طبق جداول زیر تنظیم کنید.

جدول ۱۷-۶- ویژگی های برجسب

Label		
ویژگی	مقدار	
Name	flowLbl	
Text		
BackColor	رنگ دلخواه	
Location	X	180
	Y	9

جدول ۱۸-۶- ویژگی های فرم

Form		
ویژگی	مقدار	
Name	Form1	
Text	جابه جایی برجسب	
Size	Width	452
	Height	362

۴- تنظیم ویژگی های دکمه ها : چهار کنترل دکمه را به فرم اضافه کرده و ویژگی های آنها را طبق جداول زیر تنظیم کنید.

جدول ۱۹-۶- ویژگی های دکمه left

Button	
ویژگی	مقدار
Name	right
Width	43
Height	28
Font	Wingdings
Text	←

جدول ۲۰-۶- ویژگی های دکمه right

Button	
ویژگی	مقدار
Name	left
Width	43
Height	28
Font	Wingdings
Text	→

جدول ۲۱-۶- ویژگی های دکمه down

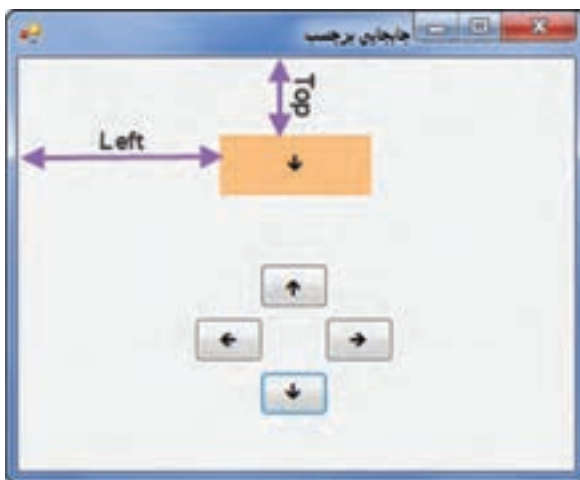
Button	
ویژگی	مقدار
Name	down
Width	43
Height	28
Font	Wingdings
Text	↓

جدول ۲۲-۶- ویژگی های دکمه up

Button	
ویژگی	مقدار
Name	up
Width	43
Height	28
Font	Wingdings
Text	↑

برای قرار دادن علامت جهت روی دکمه‌ها می‌توان از تصویر استفاده کرد و ویژگی BackgroundImage دکمه را تغییر داد و یا از نمادها و ویژگی Text دکمه استفاده کرد. در اینجا ما از روش دوم استفاده می‌کنیم. ویژگی Font دکمه‌ها را مساوی Wingdings قرار می‌دهیم. در این فونت نمادها، دارای کد اسکی ۲۳۱، ۲۳۲، ۲۳۳ و ۲۳۴ هستند. برای قرار دادن این نمادها در ویژگی Text هر دکمه، کلید Alt را نگه داشته و کد اسکی کلید را با کلیدهای ماشین حساب صفحه کلید وارد کنید.

۵- نوشتن رویداد click برای دکمه‌ها : در زمان طراحی فرم موقعیت هر کنترل نسبت به فرمی که روی آن قرار دارد با ویژگی Location که دارای دو مشخصه X و Y است تعیین می‌شود ولی برای تغییر موقعیت کنترل در زمان اجرا از دو ویژگی Left و Top که به ترتیب معادل X و Y هستند استفاده می‌کنیم. بنابراین مشخصه Left، فاصله کنترل از سمت چپ فرم و مشخصه Top، فاصله کنترل از بالای فرم است (شکل ۱۶-۶). بنابراین هر چه به مقدار مشخصه Left اضافه شود، فاصله کنترل از سمت چپ فرم بیشتر می‌شود و کنترل به سمت راست جابه‌جا می‌شود. هر چه از مقدار مشخصه Left کم شود، فاصله کنترل از سمت چپ فرم کمتر می‌شود و کنترل به سمت چپ جابه‌جا می‌شود. به همین ترتیب هر چه به مقدار مشخصه Top اضافه شود، فاصله کنترل از بالای فرم بیشتر می‌شود و کنترل به سمت پایین جابه‌جا می‌شود. هر چه از مقدار مشخصه Top کم شود، فاصله بالای کنترل از بالای فرم بیشتر می‌شود و کنترل به سمت بالا جابه‌جا می‌شود.



شکل ۱۶-۶- ویژگی Top و Left کنترل

روی هر یک از دکمه‌ها دابل کلیک کرده، تا الگوی رویداد کلیک آن نوشته شود، سپس رویداد click دکمه‌ها را به صورت زیر بنویسید :

```
private void right_Click(object sender, EventArgs e)
{
    flowLbl.Left += 10;
    flowLbl.Text = right.Text;
}
```

```
private void up_Click(object sender, EventArgs e)
{
    flowLbl.Top -= 10;
    flowLbl.Text = up.Text;
}
```

```
private void down_Click(object sender, EventArgs e)
{
    flowLbl.Top += 10;
    flowLbl.Text = down.Text;
}
```

```
private void left_Click(object sender, EventArgs e)
{
    flowLbl.Left -= 10;
    flowLbl.Text = left.Text;
}
```

برنامه ۴-۶

در کد نوشته شده، برچسب به هر جهتی حرکت کند آن جهت روی برچسب نشان داده می‌شود. به این منظور با کلیک هر دکمه مقدار ویژگی Text آن دکمه را در مشخصه Text برچسب قرار داده‌ایم.

توسعه و بهبود برنامه : با اجرای برنامه متوجه خواهید شد که با کد نوشته شده برچسب می‌تواند

از محدوده فرم خارج می‌شود. حداکثر مقداری که می‌توان در دو ویژگی Top و Left برچسب قرار داد تا از فرم خارج نشود چقدر است؟ کد برنامه را تغییر دهید تا با تغییر مقدار این دو مشخصه، برچسب از محدوده فرم خارج نشود.

کار در کارگاه ۶: مدس تصویر

مثال ۶-۶: کنترل جعبه متن^۱

در این مثال با ایجاد فرمی مانند شکل ۶-۱۷ یک بازی می‌سازیم که در آن کاربر یک تصویر پنهان را حدس بزند! بازیکن برای اینکه بتواند تصویر مورد نظر را حدس بزند باید روی بخش‌هایی از تصویر کلیک کند تا آن بخش نمایش داده شود. هر زمان که کاربر تصویر را تشخیص داد، باید تشخیص خود را در جعبه متن بنویسد. در صورت درست بودن تشخیص کاربر، امتیاز او نمایش داده می‌شود.



شکل ۶-۱۷- فرم بازی حدس تصویر

الگوریتم یا روش انجام کار: برای اینکه عملیات نمایش قسمتی از تصویر را بتوانیم شبیه‌سازی کنیم در این برنامه از یک روش ساده اما کارا استفاده می‌کنیم. سطح تصویر را با چندین Label می‌پوشانیم. با کلیک هر برچسب، آن برچسب پنهان خواهد شد و قسمتی از تصویر که زیر آن قرار دارد نمایش داده می‌شود.

روش امتیاز دهی را نیز به این صورت پیاده‌سازی می‌کنیم که ۹ برچسب برای پنهان کردن

^۱ - Text Box

تصویر قرار می‌دهیم و به صورت زیر برای هر یک امتیازی مشخص می‌کنیم.

۵ Label1	۱۰ Label2	۵ Label3
۱۰ Label4	۱۰ Label5	۱۰ Label6
۵ Label7	۱۰ Label8	۵ Label9

شکل ۱۸-۶

دلیل آنکه به گوشه‌ها امتیاز کمتری داده‌ایم این است که معمولاً در گوشه‌های تصویر جزئیات کمتری برای شناخت آن وجود دارد. با پنهان شدن هر برجسب امتیاز آن از مجموع امتیاز (۷۰) کم می‌شود. با وارد کردن حدس داخل جعبه متن و زدن دکمه، حدس مورد نظر با جواب مقایسه می‌شود و پیام مناسب نمایش داده می‌شود.

۱- ایجاد پروژه: وارد برنامه VS شوید و پروژه جدیدی از نوع WindowsFormsApplication به نام GussePictureProj ایجاد نمایید.

۲- درج کنترل‌ها: در قسمت پایین فرم یک کنترل جعبه متن (TextBox)، یک دکمه و یک برجسب برای حدس و نشان دادن امتیاز کاربر قرار دهید. در بخش میانی یک جعبه تصویر (PictureBox) و ۹ برجسب برای پنهان کردن تصویر قرار دهید.

۳- تنظیم ویژگی‌های برجسب‌ها، جعبه متنی و دکمه: ویژگی‌های جعبه متن، دکمه و برجسب‌ها را طبق جداول زیر تنظیم کنید.

جدول ۲۳-۶ - ویژگی‌های جعبه متن

TextBox	
ویژگی	مقدار
Name	guess
Text	

جدول ۶-۲۴ - ویژگی های دکمه

Button	
ویژگی	مقدار
Name	check
Text	درست است؟

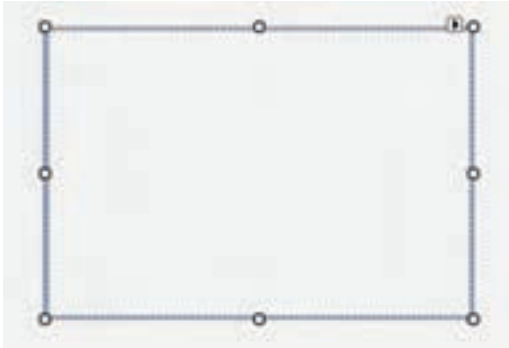
جدول ۶-۲۵ - ویژگی های برجسب امتیاز

Label (نمایش امتیاز)	
ویژگی	مقدار
Name	result
Text	حذف مقدار

جدول ۶-۲۶ - ویژگی های برجسب های پنهان کننده تصویر

Label2, Label3,, Label10 (برجسب هایی برای پنهان کردن تصویر)		
ویژگی	مقدار	توضیح
AutoSize	false	غیر فعال کردن اندازه خودکار
BackColor	#93aae1	تغییر رنگ پس زمینه
Text		متن داخل برجسب را خالی قرار می دهد.
Width	91	عرض کنترل را بر حسب پیکسل مشخص می کند.
Height	60	ارتفاع کنترل را بر حسب پیکسل مشخص می کند.

بعد از قراردادن یکی از برجسب هایی که برای پنهان کردن تصویر لازم است روی فرم در محل مناسب ، و مقداردهی ویژگی های آن براساس جدول ۶-۲۶، با نسخه برداری (copy) از آن، برجسب ها را به ۹ عدد می رسانیم. سپس این برجسب ها را به صورت دقیقی کنار هم قرار می دهیم تا در کنار هم، شکل مستطیلی را ایجاد کنند.



۴- تنظیم ویژگی‌های جعبه

تصویر: یک کنترل pictureBox به فرم اضافه می‌کنیم. این عمل را توسط کشیدن ماوس روی فرم، طوری انجام می‌دهیم که اندازه آن برابر با اندازه مستطیل ساخته شده توسط برجسب‌ها باشد.

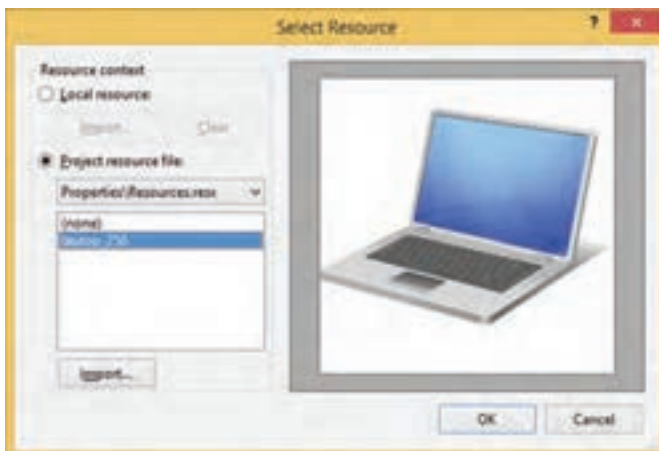
شکل ۱۸-۶ هم اندازه کردن کنترل کادر تصویر با مستطیل شامل برجسب‌ها

ویژگی‌های جعبه تصویر را مطابق جدول زیر تنظیم می‌نماییم:

جدول ۲۷-۶ ویژگی‌های جعبه تصویر

PictureBox	
ویژگی	مقدار
Name	Pic
SizeMode	StretchImage

همچنین توسط ویژگی Image، تصویر آن را مشخص می‌کنیم.



شکل ۱۹-۶ تنظیم ویژگی Image کنترل کادر تصویر

همان طور که ملاحظه می‌نمایید بعد از انتخاب عکس، به دلیل آنکه کنترل PictureBox بعد از Label ها ایجاد شده است روی آنها قرار می‌گیرد.



شکل ۲۰-۶- قرار گرفتن کادر تصویر روی برجسبها

برای آنکه کنترل PictureBox پشت برجسبها قرار گیرد، روی آن راست کلیک کرده، گزینه Send To Back را انتخاب کنید.



شکل ۲۱-۶- انتقال کنترل به پشت کنترل‌های دیگر

به این ترتیب تصویر توسط برجسبها پنهان خواهد شد.

۵- تعریف متغیر برای نگهداری امتیاز: در کلاس Form روی فرم دابل کلیک کنید تا پنجره کد باز شود.

در کلاس فرم متغیر score را تعریف می‌کنیم تا بتوانیم در تمام رویدادهای فرم از آن استفاده کنیم.

```
public partial class Form1 : Form
{
    byte score = 70;
    ....
```

این متغیر را از نوع byte در نظر گرفته ایم و آن را برابر با ۷۰ قرار داده ایم. زیرا مجموع امتیازها ۷۰ است.

سؤال: چرا برای نگهداری امتیاز کاربر متغیری از نوع Byte تعریف کردیم؟

۶- نوشتن رویداد click برای برچسب‌های پنهان‌کننده تصویر: با کلیک روی برچسب، آن برچسب پنهان می‌شود. بنابراین در رویداد کلیک هر برچسب باید آن برچسب را پنهان کرد. همان‌طور که در کارگاه ۲ دیدید، پیدا و پنهان بودن یک کنترل به یک ویژگی کنترل بستگی دارد و آن ویژگی، Visible می‌باشد. اگر مقدار این ویژگی true باشد آن کنترل نمایش داده می‌شود و اگر مقدار آن false باشد، پنهان می‌شود.

برای ایجاد رویداد کلیک برچسب، همان‌طور که قبلاً گفته شد، می‌توانید روی آن دوبار کلیک کنید و یا از لیست رویدادها استفاده کنید. بعد از ایجاد کد رویداد برای برچسب label1 خواهیم داشت:

```
private void label1_Click(object sender, EventArgs e)
{
}
```

با کلیک روی این برچسب باید این کنترل پنهان شود و ۵ امتیاز از امتیاز کل کم شود؛ زیرا همان‌طور که در طرح امتیاز گفته شد، برچسب‌های گوشه ۵ امتیاز خواهند داشت.

```
private void label1_Click(object sender, EventArgs e)
{
    score -= 5;
    label1.Visible = false;
}
```

برای سایر برچسب‌هایی که تصویر را می‌پوشانند نیز همین عملیات را انجام می‌دهیم. اما توجه داشته باشید که در رویداد کلیک هر برچسب باید همان برچسب پنهان شود. همچنین برای برچسب‌هایی که در گوشه قرار نمی‌گیرند باید ۱۰ امتیاز کسر شود. برای برچسب‌های ۳، ۷ و ۹

مانند برچسب شماره ۱، ۵ امتیاز و برای سایر برچسب‌ها ۱۰ امتیاز کم می‌شود. به عنوان مثال برای label2 داریم:

```
private void label2_Click(object sender, EventArgs e)
{
    score -= 10;
    label2.Visible = false;
}
```

کدهای مربوط به برچسب‌ها در برنامه ۵_۶ آورده شده است.

```
private void label1_Click(object sender, EventArgs e)
{
    score -= 5;
    label1.Visible = false;
}
private void label2_Click(object sender, EventArgs e)
{
    score -= 10;
    label2.Visible = false;
}
private void label3_Click(object sender, EventArgs e)
{
    score -= 5;
    label3.Visible = false;
}
private void label4_Click(object sender, EventArgs e)
{
    score -= 10;
    label4.Visible = false;
}
private void label5_Click(object sender, EventArgs e)
{
    score -= 10;
```



```

        label5.Visible = false;
    }
    private void label6_Click(object sender, EventArgs e)
    {
        score -= 10;
        label6.Visible = false;
    }
    private void label7_Click(object sender, EventArgs e)
    {
        score -= 5;
        label7.Visible = false;
    }
    private void label8_Click(object sender, EventArgs e)
    {
        score -= 10;
        label8.Visible = false;
    }
    private void label9_Click(object sender, EventArgs e)
    {
        score -= 5;
        label9.Visible = false;
    }
}

```

برنامه ۵-۶- کد مربوط به برجسب‌ها

۷- نوشتن رویداد click برای دکمه: کاربر پس از وارد کردن حدس خود در جعبه متن باید روی دکمه کلیک کند. برای بررسی اینکه حدس کاربر درست است یا خیر، رویداد کلیک دکمه را می‌نویسیم. توسط شرط (if) بررسی می‌کنیم که آیا متن درون جعبه متنی (guess) برابر متن «لپ‌تاپ» می‌باشد یا خیر. اگر شرط درست بود پیام حدس درست به همراه امتیاز در برجسب result قرار می‌گیرد و در غیر این صورت پیام حدس اشتباه در برجسب result جای می‌گیرد. لازم به یادآوری است که متد ToString مقدار متغیر score را به متن تبدیل می‌کند.

```
private void check_Click(object sender, EventArgs e)
{
    if (guess.Text == «لپ تاپ»)
        result.Text = «آفرین درست حدس زدی.. امتیاز» + score.ToString();
    else
        result.Text = «حدست درست نبود!»;
}
```

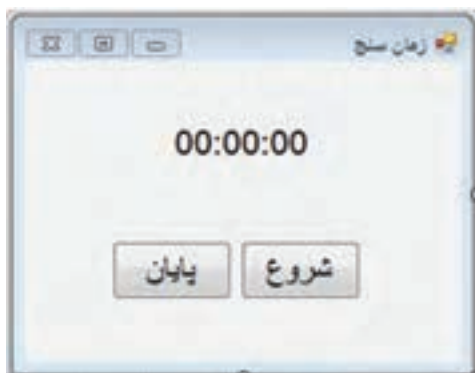
برنامه ۶-۶

توسعه و بهبود برنامه: به کدهایی که نوشته اید فکر کنید و ببینید آیا می‌توانید ایده‌های دیگری برای این بازی بدهید؟ به عنوان نمونه بیاندیشید چگونه می‌توان برای تعداد پاسخ‌های اشتباه محدودیت گذاشت. به عنوان مثال با دادن ۲ پاسخ اشتباه، پیام اتمام بازی داده شود.

کار در کارگاه ۷: طراحی زمان‌سنج دیجیتال

مثال ۷-۶: کنترل زمان‌سنج

در این مثال می‌خواهیم با طراحی فرم شکل ۶-۲۲ برنامه‌ای بنویسیم که کار زمان‌سنج را انجام دهد. الگوریتم یا روش انجام کار: این برنامه با کلیک دکمه شروع، زمان‌سنج را راه‌اندازی کرده و با کلیک دکمه پایان، زمان‌سنج را متوقف می‌کند. بنابراین به دو دکمه فرمان، یک کنترل برجسب و یک کنترل زمان‌سنج نیاز داریم.



شکل ۶-۲۲- فرم زمان‌سنج دیجیتالی

۱- ایجاد پروژه: وارد برنامه VS شوید و پروژه جدیدی از نوع WindowsFormsApplication به نام TimerProj ایجاد نمایید.

۲- درج کنترل‌ها: روی فرم دو کنترل دکمه و یک برچسب قرار دهید.

۳- تنظیم ویژگی‌های دکمه‌ها، برچسب و فرم: ویژگی‌های فرم و کنترل‌ها را طبق جدول زیر تنظیم کنید.

جدول ۶-۲۸ ویژگی‌های دکمه شروع جدول ۶-۲۹ ویژگی‌های دکمه پایان جدول ۶-۳۰ ویژگی‌های برچسب

Label	
ویژگی	مقدار
Name	timer Lbl
Text	00: 00: 00

Button	
ویژگی	مقدار
Name	stopBut
Text	پایان

Button	
ویژگی	مقدار
Name	startBut
Text	شروع

جدول ۶-۳۱ ویژگی‌های فرم

Form		
	ویژگی	مقدار
	Name	Form1
	Text	زمان سنج
	RightToLeft	yes
	RightToLeftLayout	True
Size	Width	292
	Height	228

جدول ۶-۳۲ ویژگی‌های زمان سنج

Timer	
ویژگی	مقدار
Name	Timer1
interval	1000
Enabled	false

۴- تنظیم ویژگی‌های زمان سنج: کنترل Timer را به فرم اضافه کرده و ویژگی‌های آن را طبق جدول ۶-۳۲ تنظیم کنید.

کنترل زمان سنج برای اطلاع از سپری شدن فاصله‌های زمانی یکسان به کار می‌رود. برای مثال اگر با رایانه کار می‌کنید، برای حفظ سلامتی لازم است هر نیم ساعت چند نرمش مناسب انجام دهید. ساعت را تنظیم می‌کنید که هر نیم ساعت زنگ بزند. زنگ ساعت به شما اطلاع می‌دهد که زمان نرمش رسیده است. کنترل زمان سنج این کار را برای برنامه انجام می‌دهد. فاصله زمانی در ویژگی Interval تایمر مشخص می‌شود. کنترل زمان سنج دارای رویداد Tick می‌باشد که مشابه زنگ ساعت است. این رویداد در فواصل زمانی یکسان که در ویژگی Interval کنترل مشخص می‌شود رخ می‌دهد. واحد زمانی ویژگی Interval میلی ثانیه است بنابراین اگر بخواهیم هرثانیه یکبار این رویداد رخ دهد، باید ویژگی Interval را مساوی ۱۰۰۰ قرار دهیم.

هرگاه کار شما با رایانه تمام شد، دیگر نیازی نیست که ساعت زنگ بزند؛ بنابراین زنگ ساعت را قطع می‌کنید. چگونه زنگ زمان سنج را قطع کنیم (رویداد Tick را متوقف کنیم)؟ کنترل‌ها دارای ویژگی Enabled هستند که از نوع Boolean است و هرگاه مقدار آن false شود، کنترل غیرفعال شده و به هیچ رویدادی واکنش نشان نمی‌دهد.

۵- تعریف متغیرهایی برای نگهداری ساعت، ثانیه و دقیقه: زمان سپری شده را به صورت ساعت، دقیقه و ثانیه باید نگهداری کنیم؛ بنابراین نیاز به متغیر داریم. متغیرها باید در تمام فرم قابل دسترسی باشند؛ بنابراین در کلاس Form آنها را تعریف کرده، با صفر مقداردهی می‌کنیم. روی فرم دابل کلیک کنید تا پنجره کد باز شود.

در کلاس فرم برای ثانیه، دقیقه و ساعت متغیرهایی با نام second، minute و hour و از نوع byte تعریف می‌کنیم.

```
Namespace TimerProject
{
    public partial class Form1 : Form
    {
        byte hour, second, minutes;
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

۶- نوشتن رویداد **click** برای دکمه‌ها : در رویداد **Click** دکمه شروع، کنترل زمان‌سنج را فعال می‌کنیم. برای این منظور ویژگی **Enabled** زمان‌سنج را با **true** مقداردهی می‌کنیم و متغیرهایی که مقدار ثانیه، دقیقه و ساعت را نگهداری می‌کنند، صفر می‌کنیم.

```
private void startBut_Click(object sender, EventArgs e)
{
    timer1.Enabled = true;
    hour = minute = second = 0;
}
```

برنامه ۸-۶

در رویداد **Click** دکمه پایان، کنترل زمان‌سنج را غیرفعال می‌کنیم.

```
private void stopBut_Click(object sender, EventArgs e)
{
    timer1.Enabled = false;
}
```

برنامه ۹-۶

۷- نوشتن رویداد **Tick** برای کنترل زمان‌سنج : از آنجا که فاصله زمانی رخ داده رویداد **Tick** را روی ۱ ثانیه ($Interval=1000$) تنظیم کردیم، باید در رویداد **Tick** به متغیر ثانیه یکی اضافه کنیم و اگر مقدار آن به ۶۰ رسید آن را صفر کرده و به متغیر دقیقه یکی اضافه کنیم و اگر مقدار متغیر دقیقه به ۶۰ رسید مقدار متغیر دقیقه را صفر کرده و به متغیر ساعت یکی اضافه می‌کنیم. در انتها ویژگی **text** کنترل برچسب **timerLbl** را تغییر می‌دهیم که زمان را نشان دهد.

```
private void timer1_Tick(object sender, EventArgs e)
{
    second+=1;
    if(second==60)
    {
        second=0;
        minute +=1;
    }
}
```

```

if (minute == 60)
{
    minute = 0;
    hour += 1;
}
timerLbl.Text=string.Format("{0:00}: {1:00}: {2:00}",
hour,minutes,second);
}

```

برنامه ۱۰-۶

متد Format نوع داده string، رشته را به الگوی خاص تبدیل می‌کند. در اینجا می‌خواهیم ساعت به صورت 00:00:00 نمایش داده شود. بنابراین جای مقادیر ساعت، دقیقه و ثانیه را با پارامترهای {0}، {1} و {2} در رشته مشخص کرده و قرار دادن 00: در پارامتر ساعت، دقیقه و ثانیه سبب می‌شود که به صورت دو رقمی نمایش داده شوند.

توسعه و بهبود برنامه: در برنامه دکمه سومی برای توقف و ادامه (pause و resume) قرار دهید که در صورت کلیک آن زمان‌سنج به صورت موقت متوقف شود و با کلیک مجدد آن به کار خود ادامه دهد.

کار در کارگاه ۸: تغییر عکس روی فرم در زمان اجرا

مثال ۸-۶: کنترل کادر محاوره‌ای انتخاب فایل

در این مثال می‌خواهیم با ایجاد فرم شکل ۲۳-۶ برنامه‌ای بنویسیم که کاربر بتواند یک تصویر را به وسیله کادر محاوره‌ای، انتخاب کرده و نمایش دهد.



شکل ۲۳-۶ فرم انتخاب تصویر از رایانه

الگوریتم یا روش انجام کار : می‌خواهیم برنامه‌ای بنویسیم که تصویر انتخاب شده توسط کاربر روی فرم نمایش داده شود. این برنامه با کلیک جعبه تصویر، یک جعبه انتخاب فایل را نمایش می‌دهد و با انتخاب فایل تصویر از کادر انتخاب فایل، تصویر آن در جعبه تصویر نمایش داده می‌شود.

۱- ایجاد پروژه : وارد برنامه VS شوید و پروژه جدیدی از نوع `WindowsFormsApplication` به نام `ShowPictureProj` ایجاد نمایید.

۲- درج کنترل‌ها : روی فرم یک کنترل `PictureBox` و یک `OpenFileDialog` قرار دهید.

۳- تنظیم ویژگی‌های جعبه تصویر : ویژگی‌های فرم و جعبه تصویر را طبق جدول زیر تنظیم کنید.

جدول ۳۳-۶- ویژگی‌های جعبه تصویر

PictureBox	
ویژگی	مقدار
Name	pictureBox1
BackColor	#ffc080
SizeMode	StretchImage
Width	430
Height	253

۴- تنظیم ویژگی‌های کادر محاوره‌ای انتخاب فایل : برای باز کردن فایل جدید در برنامه‌هایی مانند `word` با انتخاب گزینه `open` از منوی `file` کادر محاوره‌ای `open` باز می‌شود و به وسیله آن می‌توان فایل موردنظر را انتخاب کرد. کنترل `OpenFileDialog` کادر محاوره‌ای `open` است. این کنترل پایین فرم قرار می‌گیرد و در زمان اجرا دیده نمی‌شود. در این مثال برای انتخاب فایل تصویر به این کنترل نیاز داریم. آن را به فرم اضافه کرده و ویژگی‌های آن را طبق جدول ۳۴-۶ تنظیم کنید.

جدول ۳۴-۶- ویژگی های کادر محاوره ای OpenFileDialog

OpenFileDialog		
ویژگی	مقدار	توضیحات
Name	OpenFileDialog1	تعیین نام
Filter	Jpg File *.jpg jpeg files *.jpeg	تعیین فرمت فایل های قابل انتخاب
FileName		خالی کردن کادر نام فایل
Width	430	تعیین عرض
Height	253	تعیین کنترل

توجه کنید که می توانید توسط ویژگی Filter نوع فایل هایی که کادر محاوره ای نشان می دهد را مشخص کنید. مقداردهی این ویژگی به صورت زیر است :

.....|نوع ۲ | شرح نوع ۲ | نوع ۱ | شرح نوع ۱

مثال : Jpg Files|*.jpg|jpeg Files|*.jpeg

شکل ۲۴-۶

در این نمونه ما می خواهیم که یک تصویر را انتخاب کنیم، بنابراین نوع فایل ها را jpg و jpeg مشخص کرده ایم. در کادر محاوره ای OpenFileDialog لیست نوع فایل ها نشان داده شده به صورت شکل ۲۵-۶ است.



شکل ۲۵-۶ - انتخاب نوع فایل های تعیین شده در ویژگی Filter

۵- نوشتن رویداد click برای جعبه تصویر : در رویداد کلیک جعبه تصویر، کادر محاوره ای انتخاب فایل را باز می کنیم.


```
private void pictureBox1_Click(object sender, EventArgs e)
{
    openFileDialog1.ShowDialog();
}
```

برنامه ۱۱-۶

کادر محاوره‌ای OpenFileDialog دارای متد ShowDialog برای باز کردن کادر محاوره‌ای است.

۶- نوشتن رویداد FileOk برای کنترل کادر محاوره‌ای انتخاب فایل : می‌خواهیم در صورت انتخاب فایل از کادر محاوره‌ای (دکمه open زده شد یا فایل دوبار کلیک شد) تصویر مورد نظر در pictureBox بارگذاری شود. با انتخاب فایل در کادر OpenFileDialog نام و مسیر فایل انتخاب شده در ویژگی FileName کنترل قرار می‌گیرد و رویداد FileOk برای کنترل رخ می‌دهد. برای نوشتن رویداد FileOk، روی کنترل OpenFileDialog در محیط طراحی دوبار کلیک کنید یا با زدن دکمه Eventes در پنجره Properties (با علامت رعد و برق) از لیست رویدادهای کنترل استفاده کنید.

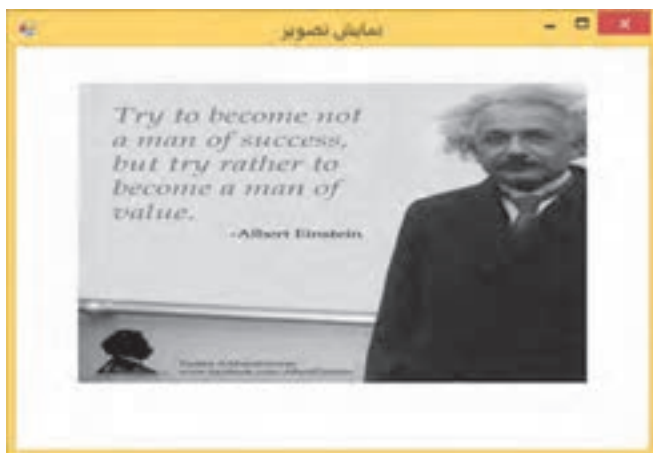
رویداد FileOk را مطابق زیر بنویسید.

```
private void openFileDialog1_FileOk(object sender, CancelEventArgs e)
{
    pictureBox1.ImageLocation = openFileDialog1.FileName;
}
```

برنامه ۱۲-۶

در این رویداد تصویر pictureBox1 بارگذاری می‌شود. بارگذاری از طریق آدرسی است که در ویژگی FileName کنترل openFileDialog1 قرار دارد. این آدرس به ویژگی ImageLocation کادر تصویر داده می‌شود. ویژگی ImageLocation آدرس یک تصویر را می‌گیرد و آن را در کادر تصویر بارگذاری می‌نماید.

توسعه و بهبود برنامه : روی فرم دکمه‌ای با عنوان تغییر تصویر قرار دهید و به جای کلیک جعبه تصویر با کلیک دکمه، تصویر جعبه تصویر را تغییر دهید.



شکل ۲۶-۶- نمایش تصویر انتخاب شده در کادر تصویر

کار در کارگاه ۹: کار با فونت و رنگ

مثال ۹-۶: کنترل کادر محاوره‌ای رنگ^۱ و کادر محاوره‌ای فونت^۲

در این مثال می‌خواهیم یک ویرایشگر متنی ساده بسازیم که در آن بتوان رنگ پس زمینه و تنظیم‌های قلم (فونت) را مشخص کرد (شکل ۲۷-۶).

برای تغییر فونت از کادر محاوره‌ای فونت و برای تغییر رنگ از کادر محاوره‌ای رنگ استفاده می‌کنیم، بنابراین در این مثال به کنترل‌های ColorDialog و FontDialog نیاز داریم.

الگوریتم یا روش انجام کار: این برنامه دارای یک جعبه متن، دو دکمه یکی برای تغییر فونت و دیگری برای تغییر رنگ قلم، یک ColorDialog و یک FontDialog است. با زدن دکمه



شکل ۲۷-۶- فرم ویرایشگر متنی ساده

قلم، کادر محاوره ای FontDialog باز می شود، فونت به وسیله کاربر انتخاب شده، روی جعبه متن اعمال می شود. با زدن دکمه «رنگ پس زمینه»، کادر محاوره ای ColorDialog باز می شود و رنگ به وسیله کاربر انتخاب شده، ویژگی رنگ پس زمینه جعبه متن تنظیم می شود.

۱- ایجاد پروژه: وارد برنامه VS شوید و پروژه جدیدی از نوع WindowsFormsApplication به نام EditorProj ایجاد نمایید.

۲- درج کنترل ها: روی فرم دو کنترل Button و یک TextBox قرار دهید.

۳- تنظیم ویژگی های دکمه ها و جعبه متن: ویژگی های جعبه متن و کنترل ها را طبق جداول زیر تنظیم کنید.

جدول ۳۷-۶- ویژگی های دکمه رنگ زمینه

Button	
ویژگی	مقدار
Name	colorBtn

جدول ۳۶-۶- ویژگی های دکمه قلم

Button	
ویژگی	مقدار
Name	fontBtn

جدول ۳۵-۶- ویژگی های جعبه متن

TextBox	
ویژگی	مقدار
Name	editor
Text	
RightToLeft	Yes
Multiline	True

در کنترل TextBox در حالت عادی تنها می توان یک خط نوشت. اما این کنترل دارای ویژگی MultiLine است که از نوع Boolean می باشد و در صورتی که مقدار آن مساوی true باشد، جعبه متن چند خطی ایجاد می کند.

۴- تنظیم ویژگی های کادرهای محاوره ای: دو کنترل FontDialog و ColorDialog را به فرم اضافه کرده و ویژگی آنها را طبق جداول زیر تنظیم کنید.

جدول ۳۸-۶- ویژگی های کادر محاوره ای رنگ جدول ۳۹-۶- ویژگی های کادر محاوره ای فونت

FontDialog	
ویژگی	مقدار
Name	fontDialog1
ShowColor	True

ColorDialog	
ویژگی	مقدار
Name	colorDialog1

ویژگی ShowColor در کادر محاوره‌ای فونت امکان انتخاب رنگ قلم را نیز می‌سازد.
۵- نوشتن رویداد click برای دکمه‌ها: در رویداد کلیک دکمه قلم، کادر محاوره‌ای تنظیمات قلم را نمایش می‌دهیم. برای این منظور متد ShowDialog() شیء fontDialog1 را فراخوانی می‌کنیم. سپس ویژگی‌های Font و ForeColor جعبه متن را با فونت و رنگ انتخاب شده در کادر محاوره‌ای فونت مقداردهی می‌کنیم.

```
private void fontBtn_Click(object sender, EventArgs e)
{
    fontDialog1.ShowDialog();
    editor.Font = fontDialog1.Font;
    editor.ForeColor = fontDialog1.Color;
}
```

برنامه ۶-۱۳



شکل ۶-۲۸- کادر محاوره‌ای فونت

در رویداد کلیک دکمه رنگ پس زمینه (colorBtn)، کادر محاوره‌ای انتخاب رنگ را نمایش می‌دهیم، بنابراین متد ShowDialog() شیء colorDialog1 را فراخوانی می‌کنیم، سپس رنگ پس‌زمینه جعبه متنی editor را با رنگ انتخاب شده مقداردهی می‌کنیم.

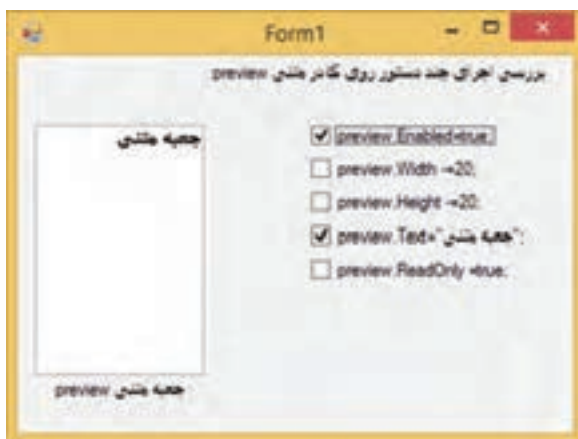
```
private void colorBtn_Click(object sender, EventArgs e)
{
    colorDialog1.ShowDialog();
    editor.BackColor = colorDialog1.Color;
}
```

برنامه ۶-۱۴

مثال ۱۰-۶: کنترل کادر علامت

در این کارگاه می‌خواهیم با نوشتن برنامه‌ای به کاربر اجازه دهیم که تغییر برخی از ویژگی‌های کنترل جعبه متن را در زمان اجرا مشاهده کند. برای دادن امکان انتخاب ویژگی‌ها به کاربر از کنترل CheckBox استفاده می‌کنیم. این کنترل در مواردی که دو حالت داریم مثل true/false یا بله / خیر کاربرد دارد. در کنار این کنترل مربع کوچکی قرار دارد که در زمان اجرا با کلیک روی آن می‌توان علامت در آن قرار داد و یا علامت آن را حذف کرد.

برنامه‌ای بنویسید که حاصل اجرای چند دستور روی یک جعبه متنی را نمایش دهد. این دستورات به صورت CheckBox قابل انتخاب باشند و حاصل اجرای آنها هم‌زمان، نمایش داده شود (شکل ۶-۲۹).



شکل ۶-۲۹- فرم اجرای هم‌زمان چند دستور روی جعبه متنی

الگوریتم یا روش انجام کار: این فرم دارای پنج CheckBox، دو Label و یک TextBox است. با توجه به CheckBox‌هایی که دارای علامت هستند، گزینه دستوری مطابق با متن آنها، روی جعبه متنی اجرا می‌شود. اگر یک گزینه علامت‌دار باشد دستور آن اجرا می‌شود و در غیر این صورت عکس آن عمل اجرا می‌شود (چرا؟).

۱- ایجاد پروژه: وارد برنامه VS شوید و پروژه جدیدی از نوع WindowsFormsApplication

به نام CheckBoxProj ایجاد نمایید.

۲- درج کنترل‌ها: روی فرم پنج کنترل CheckBox، دو Label و یک TextBox قرار دهید.

۳- تنظیم ویژگی‌های کادرهای علامت، برچسب‌ها و جعبه متنی: ویژگی‌های کادرهای علامت، برچسب‌ها و جعبه متنی را طبق جداول زیر تنظیم کنید.

جدول ۴۱-۶- ویژگی‌های کادر علامت ۲

CheckBox1	
ویژگی	مقدار
Name	setEnabled
Text	preview.Enabled=true;
RightToLeft	No
Checked	True

جدول ۴۰-۶- ویژگی‌های کادر علامت ۱

checkBox2	
ویژگی	مقدار
Name	setWidth
Text	preview.Width -= 20;
RightToLeft	No
Checked	False

کنترل CheckBox دارای ویژگی Checked است که از نوع Boolean می‌باشد و اگر مقدار آن مساوی true باشد، کنترل دارای علامت است. برای آنکه با علامت‌دار شدن CheckBoxها عملیات مورد نظر روی جعبه متن انجام شود، در ویژگی Text کنترل‌ها دستور مورد نظر قرار داده‌ایم.

جدول ۴۳-۶- ویژگی‌های کادر علامت ۳

checkBox3	
ویژگی	مقدار
Name	setHeight
Text	preview.Height -= 20;
RightToLeft	No
Checked	False

جدول ۴۲-۶- ویژگی‌های کادر علامت ۴

checkBox4	
ویژگی	مقدار
Name	setReadOnly
Text	preview.ReadOnly= true;
RightToLeft	No
Checked	False

جدول ۴۴-۶ ویژگی های جعبه متن

TextBox 1	
ویژگی	مقدار
Name	preview
Text	جعبه متنی
RightToLeft	Yes
Multiline	True

جدول ۴۵-۶ ویژگی های کادر علامت ۵

checkBox5	
ویژگی	مقدار
Name	setText
Text	preview.Text = "جعبه متنی"
RightToLeft	No
Checked	True

دو برچسب روی فرم داریم که یکی عنوان پروژه را نشان می دهد و دیگری عنوانی برای جعبه متنی مشخص می کند. ویژگی این برچسب ها مطابق جداول زیر است.

جدول ۴۶-۶ ویژگی برچسب ۲

label 2	
ویژگی	مقدار
Name	label2
Text	جعبه متنی preview

جدول ۴۷-۶ ویژگی برچسب ۱

label 1	
ویژگی	مقدار
Name	label1
Text	بررسی اجرای چند دستور روی کادر متنی preview

۴- نوشتن رویداد **CheckedChanged** برای کادرهای علامت: برای آنکه با علامت دار شدن یک **CheckBox** عملیات مورد نظر روی جعبه متن انجام شود، در ویژگی **Text** کنترل ها دستور مورد نظر را قرار داده ایم، بنابراین کافی است همان دستور را در کد بنویسیم. در چه رویدادی؟ با کلیک کنترل **CheckBox**، کنترل علامت دار شده یا علامت آن حذف می شود و برای کنترل رویداد **CheckedChanged** رخ می دهد؛ بنابراین در هنگام رخ دادن این رویداد، ابتدا باید بررسی کنیم که اگر **CheckBox** مورد نظر انتخاب شده (علامت دار) دستور ذکر شده در متنش را انجام دهیم و در غیر این صورت باید عملی عکس آن انجام شود تا جعبه متنی به حالت قبش برگردد. پس از آنکه رویداد **CheckedChanged** را برای همه **CheckBox** ها ایجاد کردیم کد آنها را به صورت برنامه ۱۵-۶ تکمیل کنید.

```
private void setWidth_CheckedChanged(object sender, EventArgs e)
{
    if (setWidth.Checked)
        preview.Width -= 20;
    else
        preview.Width += 20;
}
```

برنامه ۱۵-۶

در این رویداد با کمک ویژگی Checked کنترل، بررسی می‌شود که کادر علامت setWidth انتخاب شده است یا خیر؟ اگر جواب درست بود از عرض جعبه متنی ۲۰ پیکسل کم می‌شود و در غیراین صورت عکس این عمل انجام می‌شود. یعنی ۲۰ واحد به عرض جعبه متنی اضافه می‌گردد. (ویژگی Width پهنای کنترل و ویژگی Height ارتفاع کنترل‌ها را برحسب پیکسل مشخص می‌کند.)

```
private void setHeight_CheckedChanged(object sender, EventArgs e)
{
    if (setHeight.Checked)
        preview.Height -= 20;
    else
        preview.Height += 20;
}
```

برنامه ۱۶-۶

در این رویداد با کمک ویژگی Checked کنترل، بررسی می‌شود که کادر علامت setHeight انتخاب شده است یا خیر؟ اگر جواب درست بود از ارتفاع جعبه متنی ۲۰ پیکسل کم می‌شود و در غیر این صورت عکس این عمل انجام می‌شود. یعنی ۲۰ واحد به ارتفاع جعبه متنی اضافه می‌گردد.

```
private void setEnabled_CheckedChanged(object sender, EventArgs e)
{
    preview.Enabled = setEnabled.Checked;
}
```

برنامه ۱۷-۶

ویژگی Enabled کنترل جعبه متن مانند ویژگی Checked کنترل کادر علامت از نوع Boolean است؛ بنابراین در صورت انتخاب کنترل `setEnabled` باید مقدار `Enabled` مساوی `true` و در غیر این صورت `false` شود، پس در این برنامه مقدار این دو ویژگی مساوی یکدیگر است.

```
private void setText_CheckedChanged(object sender, EventArgs e)
{
    if (setText.Checked)
        preview.Text = "جعبه متنی";
    else
        preview.Text = "";
}
```

برنامه ۶-۱۸

در این رویداد نیز با بررسی وضعیت انتخاب کادر علامت `setText`، متن جعبه متنی را برابر با «جعبه متنی» می‌کند یا آن را خالی می‌نماید.

ویژگی `ReadOnly` کنترل جعبه متن مانند ویژگی `Enabled` از نوع Boolean است و اگر مساوی `true` باشد جعبه متن برای کاربر مانند کنترل برچسب می‌شود و کاربر نمی‌تواند چیزی در آن بنویسد.

```
private void setReadOnly_CheckedChanged(object sender, EventArgs e)
{
    preview.ReadOnly = setReadOnly.Checked;
}
```

برنامه ۶-۱۹

خودآزمایی فصل ششم

۱- هر ویژگی از سمت راست را با توضیح سمت چپ مطابقت دهید(یک توضیح اضافی است).	
۱- تعیین نوع و اندازه قلم متن یک شیء	Visible
۲- رنگ متن روی شیء	Text
۳- متن نمایش داده شده روی شیء	Name
۴- موقعیت شیء نسبت به گوشه بالا و سمت چپ فرم	RightToLeft
۵- تثبیت فاصله لبه‌های شیء از لبه‌های فرم	Font
۶- پشتیبانی شیء از زبان‌های راست به چپ	Anchor
۷- پنهان کردن شیء	ForeColor
۸- نام شیء برای دسترسی به ویژگی‌ها و متدهایش	location
۹- فعال کردن شیء	

۲- در پنجره Properties، چطور متوجه می‌شویم که کدام شیء یا کنترل در روی فرم، انتخاب شده است؟

۳- لیستی از کنترل‌هایی که تاکنون با آنها آشنا شدید را روی یک طرف کاغذی به ابعاد ۱۰×۵ سانتی‌متر بنویسید و در طرف دیگر کاربرد آن را ذکر کنید. سپس فیش‌های نوشته شده را با هم کلاسی خود به اشتراک بگذارید.

۴- فیش نویسی سؤال ۳ را برای کنترل‌ها و ویژگی‌های آن انجام دهید. در یک طرف کاغذ نام کنترل و طرف دیگر، سه تا چهار مورد ویژگی‌هایی که کاربردی هستند را بنویسید. می‌توانید فیش‌ها را در کلاس به کمک معلم خود، در یک کارگروهی و مسابقه تمرین کنید.

تمرینات برنامه نویسی فصل ششم

۱- یک فرم مطابق با یک صفحه کلید بدون واکنش به رویدادها طراحی کنید.



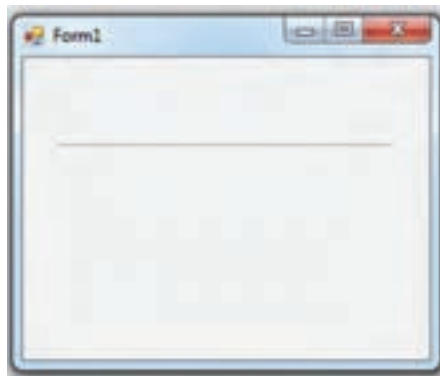
۲- یک فرم مطابق با کنترل تلویزیون بدون واکنش به رویدادهای کلیک ماوس بسازید. احتمالاً این کنترل باطری ندارد!



۳- یک فرم شامل چهار تصویر که هر یک مربوط به یک فصل باشد، بسازید. در زیر هر یک از تصاویر، یک برجسب قرار داده و نام فصل مربوطه را بنویسید. طراحی صفحه را با دقت و با رعایت فاصله یکسان و مناسب تصاویر با یکدیگر و همچنین از لبه‌های فرم انجام دهید. از ابزارهای منوی

- Format و یا در نوار ابزار، برای تنظیم موقعیت تصاویر و برچسب‌ها استفاده کنید.
- ۴- شبیه تمرین قبل را با موضوع تیم‌های ورزشی مورد علاقه خود بسازید. حداقل دو تیم ورزشی را در فرم معرفی کنید و لوگو تیم نشان داده شود.
- ۵- یک فرم شامل یک تصویر، یک جعبه متن و سه کادر علامت برای معرفی یکی از جاذبه‌های گردشگری شهر خود طراحی کنید. توضیحات آن جاذبه را در جعبه متن قرار دهید. سپس برنامه‌ای بنویسید که کاربر بتواند از طریق انتخاب یا عدم انتخاب کادرهای علامت، توضیحات را به زبان انگلیسی یا فارسی نمایش دهد و رنگ قلم و زمینه جعبه متن را تغییر دهد.
- ۶- تمرین زیر که به زبان انگلیسی ارائه شده است را بخوانید و سپس انجام دهید.

This example shows how to create a bevel line in Windows Forms. This line can be used as a visual separator of controls on a form.



To simulate the line in Windows Forms use a Label control. Set its Height to 2 pixels and BorderStyle to Fixed3D. That's all.

Now, create a new Windows Forms Application project and add a Label to the Form. Set its properties as shown in below table. Now, run the program.

Label Control	
Property	Value
AutoSize	False
BorderStyle	Fixed3D
Height	2
Width	250

The **OpenFileDialog** component allows users to browse the folders of their computer or any computer on the network and select one or more files to open. The dialog box returns the path and name of the file the user selected in the dialog box.

Use the **ShowDialog** method to display the dialog box and the OpenFile method to open the file.

واژگان و اصطلاحات انگلیسی فصل ششم

ردیف	واژه انگلیسی	معنی واژه به فارسی
۱	Anchor	
۲	Button	
۳	Button Control	
۴	CheckBox	
۵	Double Click	
۶	Interval	
۷	ReadOnly	
۸	Tulip	



تبدیل ضمنی انواع داده به یکدیگر در C#

گاهی هنگام استفاده از متغیرها و مقداردهی به آنها، مقداری که به متغیر منتسب می‌کنیم با نوع متغیر یکسان نیست اما سی شارپ پیام خطا نمی‌دهد. چرا؟
 زیرا سی شارپ در صورتی که امکانش وجود داشته باشد به صورت ضمنی، نوع داده آن مقدار را به نوع داده متغیر تبدیل می‌کند. در جدول زیر مشخص شده که سی شارپ هر نوع داده را به کدام یک از انواع داده می‌تواند تبدیل کند.

ردیف	نوع داده	انواع داده قابل تبدیل
۱	<u>s</u> byte	short, int, long, float, double, or decimal
۲	<u>b</u> yte	short, ushort, int, uint, long, ulong, float, double, or decimal
۳	<u>s</u> hort	int, long, float, double, or decimal
۴	<u>u</u> short	int, uint, long, ulong, float, double, or decimal
۵	<u>i</u> nt	long, float, double, or decimal
۶	<u>u</u> int	long, ulong, float, double, or decimal
۷	<u>l</u> ong	float, double, or decimal
۸	<u>c</u> har	ushort, int, uint, long, ulong, float, double, or decimal
۹	<u>f</u> loat	double
۱۰	<u>u</u> long	float, double, or decimal

تبدیل صریح (casting) انواع داده به یکدیگر در C#

گاهی هنگام انتساب یک عبارت یا مقدار به یک متغیر سی شارپ خطایی مبنی بر اینکه نمی تواند نوع داده مقدار یا عبارت را به نوع داده متغیر تبدیل کند می دهد. برای حل این مشکل می توان از تبدیل صریح نوع داده استفاده کرد. در جدول زیر مشخص شده که با روش تبدیل صریح هر نوع داده را به کدام یک از انواع دیگر داده می توان تبدیل کرد.

جدول تبدیل صریح (casting) انواع داده به یکدیگر

ردیف	نوع داده	انواع داده قابل تبدیل
1	sbyte	byte , ushort, uint, ulong, or char
2	byte	sbyte or char
3	short	sbyte , byte, ushort, uint, ulong, or char
4	ushort	sbyte , byte, short, or char
5	int	sbyte , byte, short, ushort, uint, ulong, or char
6	uint	sbyte , byte, short, ushort, int, or char
7	long	sbyte , byte, short, ushort, int, uint, ulong, or char
8	char	sbyte , byte, or short
9	float	sbyte , byte, short, ushort, int, uint, long, ulong, char, or decimal
10	ulong	sbyte , byte, short, ushort, int, uint, long, or char
	double	sbyte , byte, short, ushort, int, uint, long, ulong, char, float, or decimal
	decimal	sbyte , byte, short, ushort, int, uint, long, ulong, char, float, or double

الگوی نمایش اطلاعات

در جدول زیر لیستی از تعیین کننده‌های فرمت (format specifiers) که در الگوی نمایش {تعیین کننده فرمت : عدد} یا الگوی نمایش {تعیین کننده فرمت : عدد تراز، عدد} برای قالب بندی اعداد به کار می‌روند، آمده است :

Format	Specifier
مقدار پولی خاص یک محل	C
انواع صحیح (integer)	D
نماد علمی	E
نقطه اعشار ثابت	F
اعداد عمومی	G
نقطه اعشار ثابت با جدا کننده کاما	N
اعداد دارای درصد	P
هگزادسیمال	X

تأثیر دقت یک قالب خاص بستگی به تنظیمات منطقه‌ای دارد. به عنوان مثال قالب پولی C، به طور خودکار قالب پولی منطقه انتخاب شده را نشان می‌دهد. برای اکثر کاربران این اطلاعات به طور پیش فرض مربوط به منطقه و زبان آنها می‌باشد.

قالب بندی سفارشی اعداد

سی‌شارپ به شما اجازه قالب بندی اعداد به صورت سفارشی را نیز می‌دهد. برای این کار کاراکترهای خاصی در اختیار شما می‌گذارد که در جدول صفحه بعد لیست آنها را مشاهده می‌کنید :

معنی	کاراکتر
عدد	#
اعشار	.
جدا کننده سه رقمی	,
درصد	%
اضافه کردن صفر	0
اعداد مثبت، منفی و مقادیر صفر را با قالب‌های خاصی نمایش می‌دهد.	;
نماد علمی	E0 E+0 E-0 e0 e+0 e-0

نقطه مکان اعشار را مشخص می‌کند. علامت # می‌تواند در برگیرنده عددی باشد که قرار است در سمت چپ یا راست آن اعشار قرار بگیرد. اگر این علامت در سمت راست اعشار قرار بگیرد دقت اعشار را مشخص می‌کند و ممکن است در صورت لزوم عدد را رند کند و اگر در سمت چپ ممیز باشد نماینده قسمت صحیح عدد می‌باشد. اگر طول عدد از تعداد علامت‌های # که در سمت چپ علامت اعشار قرار دارند بیشتر باشد، کل عدد نشان داده می‌شود.

کاراکتر E و e برای نمایش اعداد به صورت نماد علمی به کار می‌روند. حداقل یک صفر و در صورت لزوم تعداد بیشتری صفر بعد از E و e می‌آیند. صفرها تعداد ارقام اعشار را نشان می‌دهند. استفاده از حرف E و یا e ممکن است باعث چاپ این حروف در خروجی شوند. در نتیجه برای تعریف توان مثبت و منفی می‌توان از علامت - و + بعد از این دو کاراکتر به صورت E+, E-، e+، e- استفاده کرد. علامت ";" هم شما را قادر می‌سازد که اعداد مثبت، منفی و مقادیر صفر را با قالب‌های خاصی جدا سازید.

پیوست ۳

متدهای کلاس String

نوع برگشتی	شرح کار متد	متد
string	ایجاد یک کپی از رشته	Clone()
bool	مقایسه دو رشته	CompareTo()
bool	جستجوی یک کاراکتر یا یک رشته	Contains()
bool	آیا رشته به کاراکتر مورد نظر ختم می‌شود؟	EndsWith()
bool	مقایسه دو رشته	Equals()
int	موقعیت وجود یک کاراکتر در رشته را بر می‌گرداند.	IndexOf()
string	تبدیل تمام کاراکترهای یک رشته به حروف کوچک	ToLower()
string	تبدیل تمام کاراکترهای یک رشته به حروف بزرگ	ToUpper()
string	درج یک کاراکتر یا یک رشته در درون یک رشته دیگر	Insert()
int	موقعیت آخرین تکرار یک کاراکتر در درون یک رشته	LastIndexOf()
int	تعداد کاراکترهای یک رشته را بر می‌گرداند.	Length
string	حذف کاراکترهای یک رشته تا یک مکان مشخص	Remove()
string	یک کاراکتر در یک رشته را با کاراکتر یا یک رشته دیگری عوض می‌کند.	Replace()
String[]	یک رشته را به چند قسمت تقسیم می‌کند.	Split()
bool	آیا رشته با یک کاراکتر خاص شروع می‌شود؟	StartsWith()
string	یک قسمت از یک رشته را جدا می‌کند و در اختیار قرار می‌دهد.	Substring()
string	فاصله و کاراکترهای Tab را از ابتدا و انتهای رشته حذف می‌کند.	Trim()