فصل دوم

رویدادهای ماوس و صفحه کلید

در بیشتربرنامه‌های گرافیکی، از ماوس برای رسم اشکال و یا انتخاب ابزارهای گرافیکی موجود در برنامه استفاده می‌شود. برای اینکه بتوانیم برنامه‌های گرافیکی کاربردی بنویسیم، لازم است رویدادهای ماوس را بشناسیم. در کتاب برنامه سازی دو با رویداد کلیک ماوس در برنامه آشنا شده و از آن استفاده کردیم، علاوه بر ماوس، واکنش‌های برنامه نسبت به کلیدهای صفحه کلید قابل کنترل و تنظیم است. در این بخش با رویدادهای دیگری از ماوس و صفحه کلید آشنا می‌شویم.

پس از پایان این فصل انتظار می‌رود که فراگیر بتواند :

۱ــ رویدادهای مربوط به ماوس و صفحه کلید را توضیح دهد.

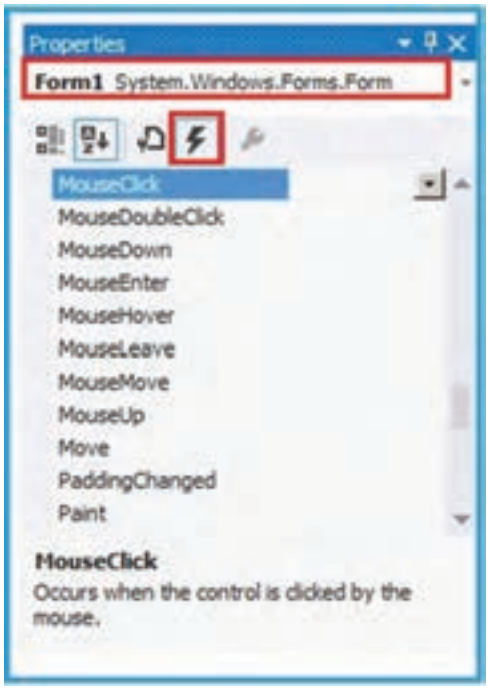
۲ــ از رویدادهای مربوط به ماوس در برنامه‌های کاربردی تعاملی[۱] استفاده نماید.

۳ــ رویدادهای مربوط به صفحه کلید را در برنامه‌های خود به کار بندد.

۱ــ Interactive

۱ــ۲ــ رویدادهای ماوس

اگر یک پروژه WFA ایجاد کنید و در پنجره Properties در قسمت Event مربوط به فرم، لیست رویدادها را مشاهده کنید، در بین آنها چند رویداد مربوط به ماوس است (شکل ۱ــ۲). عملکرد هر رویداد در جدول ۱ــ۲ آورده شده است.



شکل ۱ــ۲ــ رویدادهای ماوس

جدول ۱ــ۲

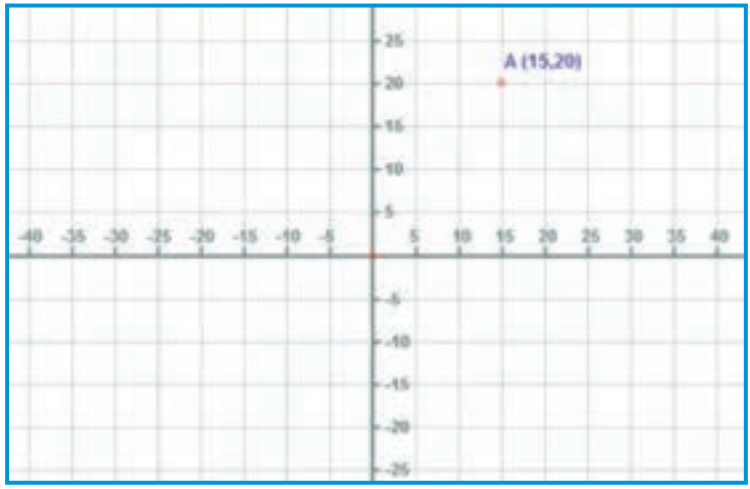
ردیف	نام رویداد	شرح رویداد
۱	MouseClick	کلیک ماوس روی فرم یا کنترل
۲	MouseDoubleClick	دابل کلیک ماوس روی فرم یا کنترل
۳	MouseDown	فشردن دکمه ماوس روی فرم یا کنترل
۴	MouseEnter	ورود از خارج فرم یا کنترل به روی آن
۵	MouseHover	چند لحظه نگه داشتن ماوس روی فرم یا کنترل
۶	MouseLeave	خارج شدن ماوس از روی فرم یا کنترل
۷	MouseMove	حرکت یا تغییر مکان ماوس روی فرم یا کنترل
۸	MouseUp	رها کردن دکمه فشرده شده ماوس

هنگامی که به وسیلهٔ ماوس بر روی فرم کلیک می‌کنید رویداد MouseClick و درصورتی که ماوس را حرکت دهید، رویداد MouseMove رخ می‌دهد. مثال زیر را برای آشنایی با رویداد MouseMove ببینید.

مثال ۱ــ۲ : برنامه‌ای بنویسید که با حرکت ماوس بر روی فرم، موقعیت ماوس با دو عدد طول و عرض نشان داده شود.

شکل ۲ــ۲ــ مختصات ماوس

الگوریتم یا روش انجام کار : از درس ریاضیات به خاطر دارید که هر نقطه در صفحه با دو عدد طول و عرض، نسبت به مبدأ نشان داده می‌شود. به آن دو عدد، **مختصات نقطه**[1] می‌گویند. شکل۳ــ۲ موقعیت نقطهٔ A در صفحه را نشان می‌دهد.



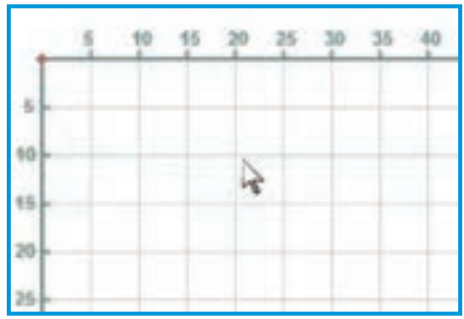
شکل ۳ــ۲ــ موقعیت نقطه در صفحه

۱ــ Coordinates of a point

نکته

در VS نقطه مبدأ برای تعیین مختصات یک شیء یا ماوس بر اساس گوشهٔ بالا سمت چپ در نظر گرفته می‌شود (شکل ۴ــ۲).

موقعیت ماوس در روی فرم با دو عدد سنجیده می‌شود. عدد اول که با حرف X نشان می‌دهیم طول است و عبارت است از فاصله ماوس تا سمت چپ فرم وعدد دوم، فاصله ماوس تا بالای فرم را نشان می‌دهد که به آن عرض نقطه گفته می‌شود و با حرف Y نشان می‌دهیم. معمولاً این دو عدد برحسب پیکسل[۱] است.



شکل ۴ــ۲ــ مبدأ مختصات در صفحه

سؤال: مقدار X و Y اشاره گر ماوس در شکل ۴ــ۲ چقدر است؟

هنگامی که ماوس را حرکت می‌دهید، دو عدد X و Y به وسیلهٔ رویداد MouseMove در اختیار ما قرار می‌گیرد. بنابراین کافی است آنها را در کنترل‌های برچسب قرار دهیم.

کار در کارگاه ۱: نمایش موقعیت (مختصات) ماوس

برای نوشتن برنامه، مراحل زیر را دنبال می‌کنیم :

۱ــ ایجاد یک پروژه ویندوزی : در برنامه VS یک پروژه جدید از نوع Windows Form Application با نام MouseEventDemo در مسیر مشخصی بسازید.

۲ــ تعیین ویژگی‌های فرم : ویژگی‌های فرم را به صورت جدول۲ــ۲ تنظیم کنید.

۱ــ Pixel

جدول ۲ــ۲ــ ویژگی‌های فرم

Form		
ویژگی		مقدار
Name		Form[1]
Text		موقعیت ماوس
Size	Width	300
	Height	300

۳ــ تعیین ویژگی‌های برچسب‌ها : دو کنترل برچسب بر روی فرم قرار دهید و ویژگی آنها را مطابق جداول زیر تعیین کنید.

جدول ۳ــ۲ــ ویژگی‌های کنترل برچسب اول

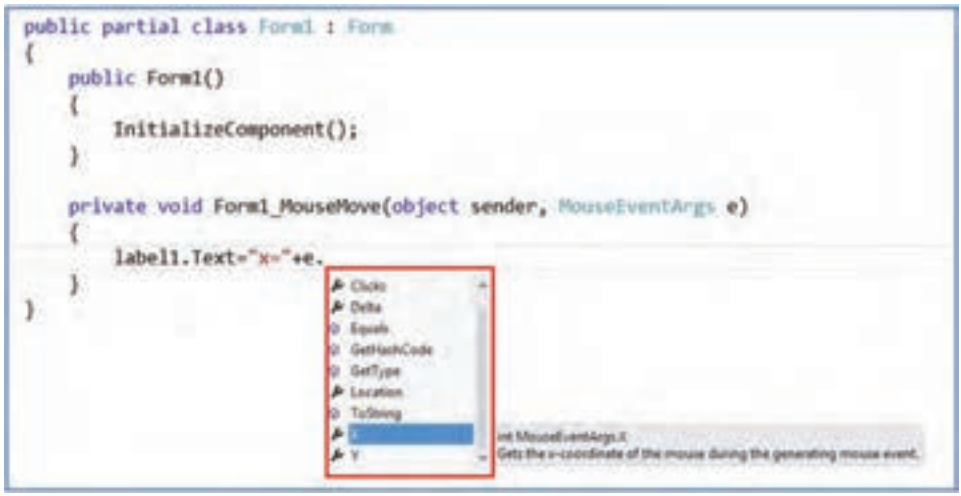
Label		
ویژگی		مقدار
Name		Label1
Text		
AutoSize		True
Location	X	70
	Y	240

جدول ۴ــ۲ــ ویژگی‌های کنترل برچسب دوم

Label		
ویژگی		مقدار
Name		Label2
Text		
AutoSize		True
Location	X	150
	Y	240

۴ــ ایجاد واکنش نسبت به حرکت ماوس : اکنون باید متدی برای واکنش نسبت به رویداد حرکت ماوس بنویسیم. برای این منظور روی فرم، کلیک کنید تا انتخاب شود. سپس در پنجره Properties مربوط به فرم، لیست رویدادها را ظاهر کنید و سراغ رویداد MouseMove بروید. روی آن دو بار کلیک کنید تا پنجره کدنویسی مربوط به متد EH رویداد مورد نظر، ظاهر شود. در داخل متد باید مقدار طول موقعیت ماوس را به وسیلهٔ کنترل برچسب label1 نشان دهیم.

به وسیلهٔ پارامتر e در متد MouseMove، می‌توانیم به جزئیات رویداد ماوس دسترسی داشته باشیم مثلاً می‌توانیم موقعیت ماوس را ببینیم. شکل ۵ــ۲ منوی IntelliSense ویژگی‌های ماوس را نشان می‌دهد. ویژگی X مقدار طول و ویژگی Y مقدار عرض موقعیت ماوس را در اختیار برنامه نویس قرار می‌دهد.



شکل ۵ــ۲

بنابراین دستورات نمایش موقعیت ماوس را مانند زیر می‌نویسیم :

```
private void Form1_MouseMove (object sender, MouseEventArgs e)
{
    label1. Text = "x=" + e.X;
    label2.Text = "y=" + e.Y;
}
```

اکنون می‌توانید برنامه را اجرا کنید و با حرکت ماوس، موقعیت آن را به صورت دو عدد مشاهده کنید.

در کتابخانه .NET کلاسی به نام Cursors تعریف شده که دارای تعدادی ویژگی استاتیک است که هریک از آنها، شکل مختلفی از علامت ماوس را در اختیار قرار می‌دهد. به‌عنوان نمونه، ویژگی Cross شکل ظاهری یا علامتی از ماوس، به‌صورت دو خط متقاطع[۱] (به شکل +) فراهم می‌کند.

۱ــ Cross

از طرفی هر کنترل از جمله فرم، دارای ویژگی به نام Cursor است که مقداردهی آن به یکی از ویژگی‌های کلاس Cursors، سبب می‌شود با قرار گرفتن ماوس در روی آن کنترل یا فرم، علامت ماوس تغییر پیدا کرده و به شکل تعیین شده ظاهر شود. مثلاً اگر می‌خواهید شکل ظاهری ماوس در هنگام حرکت بر روی فرم، به صورت علامت دو خط متقاطع (+) دیده شود، باید ویژگی Cursor فرم را به صورت زیر تغییر دهید :

```
this.Cursor = Cursors.Cross;
```

از کلمه رزرو شده this برای دسترسی به شیء فرم استفاده شده است.

به وسیلهٔ منوی IntelliSense می‌توانید از دیگر ویژگی‌های کلاس Cursors مطلع شوید. اکنون به متد EH مربوط به حرکت ماوس، دستور تغییر شکل ماوس را مانند شکل ۶ــ۲ اضافه می‌کنیم.

شکل ۶ ــ۲ــ منوی IntelliSense کلاس Cursors

```
private void Form1_MouseMove (object sender, MouseEventArgs e)
{
    label1. Text = "x=" + e.X;
    label2.Text = "y=" + e.Y;
    this.Cursor = Cursors. Cross;
}
```

برنامه را مجدداً اجرا کنید و به شکل ظاهری ماوس توجه کنید. در صورتی که بخواهیم شکل ظاهری ماوس، به حالت اولیه (معمولاً علامت فلش) برگردد، از ویژگی Default کلاس Cursors

مطابق دستور زیر استفاده می‌کنیم.

```
this.Cursor = Cursors.Default;
```

سؤال: زمانی که برای فرم، شکل ماوس را تغییر می‌دهید، شکل ماوس برای کدام کنترل‌های روی فرم تغییر می‌کند؟ برای کدام نوع از کنترل‌ها تغییر نمی‌کند؟

کاری که در بالا انجام دادیم را می‌توان در زمان طراحی انجام داد تا خود ویژوال استودیو کد را ایجاد کند. کافی است فرم را انتخاب کنیم و از پنجره properties ویژگی Cursor را برابر Cross قرار دهیم.

اکنون می‌خواهیم زمانی که ماوس روی برچسب‌ها قرار می‌گیرد شکل اشاره گر ماوس به حالت عادی فلش تغییر پیدا کند. برای این کار می‌توانیم از دو روش استفاده کنیم :

۱ــ نوشتن کد به وسیله VS : اگر ویژگی Cursor هر دو برچسب را در پنجره Properties برابر Default قراردهیم کدها به وسیلهٔ VS نوشته می‌شود.

۲ــ نوشتن کد به وسیله برنامه نویس : اگر بخواهیم خودمان کد را بنویسیم به جای استفاده از پنجره Properties دستورات زیر را در رویداد Form_Load می‌نویسیم.

```
label1.Cursor = Cursors.Default;
label2.Cursor = Cursors.Default;
```

رویداد Form_Load زمان بارگذاری فرم در حافظه رخ می‌دهد؛ بنابراین به محض ایجاد فرم شکل اشاره‌گر ماوس برای برچسب‌ها به صورت پیش فرض که همان فلش است، تغییر می‌کند.

نکته

شکل ظاهری ماوس که به وسیلهٔ ویژگی‌های کلاس Cursors فراهم می‌شود، مطابق با شکل‌های تعیین‌شده برای ماوس، در سیستم عامل است. همان‌طورکه می‌دانید به وسیلهٔ تنظیمات ماوس در کنترل‌پنل سیستم‌عامل ویندوز، کاربر می‌تواند شکل‌های استاندارد ماوس را تغییر دهد. به همین دلیل ویژگی Default که در حالت استاندارد علامت فلش است، ممکن است شکل دیگری از ماوس را فراهم

نماید که در این صورت شکل ظاهری ماوس در ویژگی‌های کلاس Cursors نیز مطابق با آن خواهد بود (شکل ۷ــ۲).

شکل ۷ــ۲

برای مطالعه

می‌توان از آرگومان ورودی دوم رویداد MouseMove و رویدادهای دیگر ماوس که ورودی از جنس MouseEventArgs دارند (مانند MouseDown و . . .) استفاده‌های مفید دیگری نیز برد :

e.Location : مکان ماوس در کنترل

e.Button : دکمه‌ای از ماوس که گرفته شده است. فشردن کلیک راست یا چپ و غیره توسط همین خصوصیت قابل شناسایی است.

e.Clicks : تعداد دفعاتی که دکمه ماوس فشرده شده و رها شده است.

کار در کارگاه ۲ــ تونل (راهرو) ماوس

مثال ۲ــ۲ : در این برنامه می‌خواهیم بازی ماز را به شکل ساده شبیه‌سازی کنیم. در فرم ما یک منطقه پیروزی وجود دارد، برای اینکه پیروز شویم باید ماوس را به آن منطقه برسانیم. اما رسیدن به این منطقه باید از مسیر خاص و امن خود باشد وگر نه بازی با شکست مواجه می‌شود.

چگونه این برنامه را طراحی کنیم؟ در این برنامه خواهید دید به راحتی، سازندهٔ یک بازی خواهید شد.

الگوریتم یا روش انجام کار : قبل از هر کاری نقشهٔ راه را پی می‌ریزیم. برای طراحی منطقه پیروزی و راهروها از Label استفاده می‌کنیم. سپس کدهایی را برای انجام عملیات مورد نیاز به برنامه اضافه می‌کنیم. این کدها شامل کد رویدادهایی است که هنگام ورود ماوس به منطقه شکست برای نمایش پیام شکست و رسیدن به منطقه پیروزی برای نمایش پیام پیروزی اجرا می‌شود.

برای نوشتن برنامه، مراحل زیر را دنبال می‌کنیم :

۱ــ ایجاد یک پروژه ویندوزی : وارد برنامه VS شوید و یک پروژه جدید از نوع Windows Form Application با نام MouseCorridor در مسیر مشخصی بسازید.

۲ــ تعیین ویژگی‌های فرم

جدول ۵ــ۲ــ ویژگی‌های فرم

Form		
ویژگی		مقدار
Name		MTunnelForm
Text		راهروی ماوس
Size	Width	400
	Height	400

۳ــ تعیین ویژگی‌های برچسب منطقه پیروزی

جدول ۶ــ۲ــ ویژگی‌های برچسب

Label	
ویژگی	مقدار
Name	goal
Text	پیروزی
AutoSize	false
BackColor	Green
ForeColor	White
TextAlign	MiddleCenter

شکل ۸ــ۲ــ تعیین ویژگی TextAlign

ویژگی TextAlign چینش متن را مشخص می‌کند. برای تغییر ویژگی TextAlign نیز به راحتی می‌توان از کادری که در اختیارمان گذاشته می‌شود استفاده نمود. با استفاده از این کادر محل متن داخل کنترل مشخص می‌شود (شکل ۸ــ۲).

پس از انجام این تنظیم‌ها می توانید اندازه برچسب را مشخص نمایید. این کار را با کشیدن گوشه‌های کنترل Label به وسیلهٔ ماوس یا تنظیم دقیق آن از طریق ویژگی‌های Width (تنظیم عرض) و Height (تنظیم ارتفاع) انجام دهید.

۴ــ تعیین ویژگی‌های برچسب راهروها

جدول ۷ــ۲ ویژگی‌های برچسب

Label	
ویژگی	مقدار
Name	Label1
Text	
AutoSize	false
BackColor	#ff 8000

با انجام چند عملیات Copy و Paste ساده، سایر برچسب‌ها را از روی برچسب اولیه نسخه‌برداری کنید و با جابه‌جا کردنشان، یک راهرو برای عبور ماوس بسازید (مانند شکل۹ــ۲).

شکل ۹ ــ۲ــ فرم بازی تونل ماوس

۵ــ ایجاد واکنش نسبت به حرکت ماوس : در این برنامه اگر ماوس از راهرو امن خود بیرون رود بازی با شکست مواجه می‌شود. رفتن از راهرو به بیرون برابر است با ورود ماوس روی فرم! بنابراین اگر رویدادی باشد که در هنگام ورود ماوس روی فرم، فراخوانی شود می‌توانیم از آن استفاده نماییم. خوشبختانه این رویداد برای کنترل‌های مختلف از جمله فرم وجود دارد. این رویداد MouseEnter نام دارد.

برای استفاده از رویداد MouseEnter در محیط طراحی می‌توانیم به این روش عمل کنیم : ابتدا روی فرم کلیک می‌کنیم تا انتخاب شود. در پنجره Properties دکمه Events را انتخاب می‌کنیم. سپس در گروه Mouse رویداد MouseEnter را پیدا می‌کنیم. با فشردن Enter (یا دو بار کلیک کردن) رویداد مورد نظر ایجاد می‌شود و ما را به محیط کد، انتقال می‌دهد.

```
private void MTunnelForm_MouseEnter(object sender, EventArgs e)
{

}
```

در این رویداد، کدی می‌نویسیم تا پیام مناسب برای شکست بازی نمایش داده شود. برای نمایش پیام از متد Show در کلاس MessageBox به صورت زیر استفاده کنید.

```
private void MTunnelForm_MouseEnter(object sender, EventArgs e)
{
    MessageBox.Show("شما بازی را باختید");
}
```

شکل ۱۰ــ۲ــ کادر پیام برنامه

اکنون نوبت آن رسیده است که عملیات اعلام پیروزی را برنامه‌ریزی کنیم. سؤال این است چه زمانی بازیکن پیروز می‌شود؟ زمانی که ماوس وارد منطقه هدف و روی برچسب goal (برچسب پیروزی) وارد می‌شود. بنابراین کلید انجام کار، استفاده از رویداد MouseEnter این کنترل خواهد بود.

این بار نیز می‌توانیم از همان روش قبل برای تعیین رویداد استفاده کنیم. اما روش دوم این است که ابتدا روی برچسب goal کلیک کنیم تا انتخاب شود. در پنجره Properties دکمه Eventes را انتخاب می‌کنیم، سپس در گروه Mouse رویداد MouseEnter را پیدا کرده و نام Victory را در آن می نویسیم. به این ترتیب نام رویداد را خودمان انتخاب نموده‌ایم. با فشردن Enter، رویداد مورد نظر با نامی که وارد کردیم، ساخته خواهد شد. با نوشتن کد پیام پیروزی، کار ما به پایان می‌رسد.

```
private void victory(object sender, EventArgs e)
{
    MessageBox.Show("تبریک..شما برنده شدید");
}
```

وقت اجرای برنامه است! و آنچه مشاهده می کنید این است که با موفقیت توانسته‌اید این بازی را بسازید. توجه کنید برای شروع بازی باید حتماً از نقطه شروع راهرو در سمت چپ فرم استفاده نمایید.

نکته

این برنامه با راهکارهای دیگر هم قابل طراحی است. به عنوان نمونه، شکست بازی به‌جای رویداد MouseEnter فرم، در رویداد MouseLeave کنترل‌های برچسب راهرو نوشته می‌شود. یا اینکه منطقه مجاز حرکت ماوس را فرم و منطقه شکست آن را راهروها درنظر گرفت.

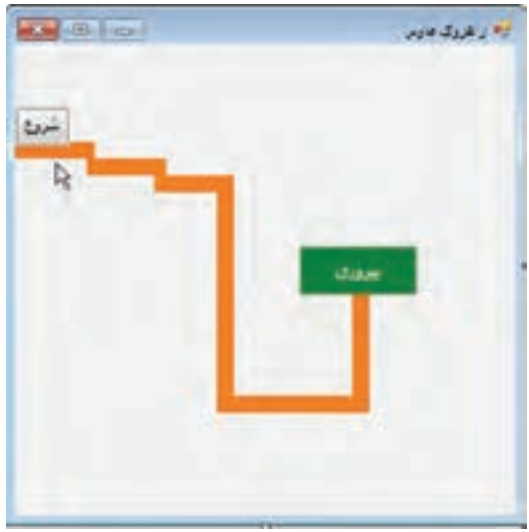
توسعه و بهبود برنامه

با انجام بازی، خواهیم دید هر ورود ماوس به فرم، ایجاد پیام شکست می‌کند. حتی اگر بازی را پیروز شده باشیم به محض آنکه دوباره ماوس را روی فرم بیاوریم پیام شکست نمایش داده می‌شود. این موضوع برای منطقه پیروزی هم صادق است. برای رفع این مشکل چه باید کرد؟ می‌توانیم از یک دکمه استفاده نماییم تا با فشردن آن بازی شروع شود و در صورتی که کاربر پیروز یا بازنده شد، روند بازی متوقف شود و پیام‌ها دیگر نمایش داده نشود. این‌کار مانند فشردن یک کلید برق می باشد. دو حالت وجود دارد : بازی در حال انجام است یا خیر!

هنگامی که از دو حالت نام می بریم می‌دانیم که می‌تواند پای یک متغیر از جنس Boolean در میان باشد! پیشنهاد این است که یک متغیر با نام play و از نوع bool در داخل کلاس فرم تعریف شود تا همه رویدادها و متدهای فرم به آن دسترسی داشته باشند.

```
public partial class MTunnelForm: Form
{
    bool play = false;
    ...
}
```

اگر این متغیر false باشد به معنای آن است که بازی شروع نشده یا تمام شده است و اگر true باشد نشان دهنده این است که بازی در حال انجام است. اکنون دکمه‌ای روی فرم با نام start ایجاد می‌کنیم و متن آن را «شروع» قرار می‌دهیم.



شکل ۱۱ ــ۲ــ فرم بازی با دکمه شروع

با فشردن دکمه، کافی است متغیر play برابر با true شود. چرا؟

```
private void start_Click(object sender, EventArgs e)
{
    play = true;
}
```

در رویداد MouseEnter برای فرم و برچسب goal نمایش پیام بستگی به مقدار متغیر play دارد؛ بنابراین، این رویدادها به صورت زیر تغییر می کند.

```
private void MTunnelForm_MouseEnter (object sender, EventArgs e)
{
    if(play)
    {
        MessageBox.Show("شما بازی را باختید");
        play = false;
    }
    }
```

```
private void victory(object sender, EventArgs e)
{
    if (play)
    {
        MessageBox.Show("تبریک.. شما برنده شدید");
        play = false;
    }

}
```

دلیل بررسی متغیر play برای اجرای دستور پیام چیست؟ این شرط بخاطر آن است که فقط اگر بازی در حالت اجرا باشد (play=true) پیام نمایش داده شود.

سؤال: چرا بعد از دستور نمایش پیام مقدار متغیر play را برابر false قرار می‌دهیم؟

۲ــ۲ــ رویدادهای صفحه کلید

هنگامی که کلیدی از صفحه کلید زده می‌شود، مانند وقتی که ماوس کلیک می‌شود، رویدادهای مختلفی رخ می‌دهد. مثلاً در هنگام فشردن و رها کردن کلیدی از صفحه کلید، سه رویداد مختلف KeyDown، KeyPress و KeyUp به ترتیب رخ می‌دهد که اطلاع می‌دهد چه کلیدی به وسیلهٔ کاربر زده شده است. در روی صفحه کلید، کلیدهایی وجود دارند که با فشردن آنها علامتی یا کاراکتری روی مانیتور ظاهر می شود، به این کلیدها یا کاراکترها، کاراکترهای چاپ شدنی[۱] می گویند. اما بعضی کلیدها نظیر Ctrl، ALT، Shift، Home، End و کلیدهای تابعی مانند F۱ تا F۱۲، علامتی را روی صفحه نشان نمی‌دهد. خوشبختانه به وسیلهٔ رویدادهای ایجاد شده، می توان از فشردن تمام کلیدها مطلع شد.

سه رویداد KeyDown، KeyPress و KeyUp به ترتیب برای اطلاع از پایین آمدن، فشردن کلید، و رها کردن کلیدی ازصفحه کلید استفاده می‌شوند.

هنگامی که رویداد KeyPress رخ می‌دهد، در متد EH مربوطه، می‌توان با استفاده از ویژگی KeyChar مطلع شد که کدام کلید به وسیلهٔ کاربر فشار داده شده است. اما از فشردن اکثر کلیدهای

۱ــ Printable Characters

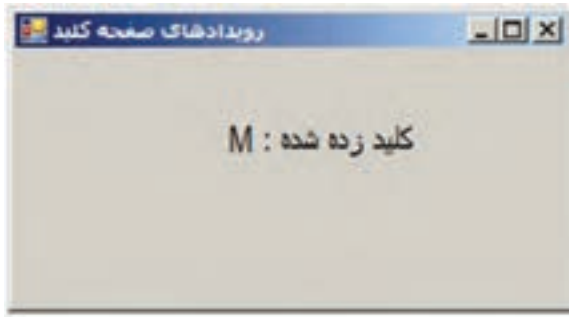
غیرچاپی نمی‌توان مطلع شد و برای این منظور در دو رویداد دیگر KeyDown و KeyUp از فشردن این کلیدها می‌توان خبردار شد.

جدول ۸ــ۲

نام رویداد	شرح رویداد
KeyPress	فشردن یک کلید چاپ شدنی
KeyDown	فشردن یا پایین رفتن کلید
KeyUp	رها کردن کلید فشرده شده

لازم به ذکر است که در زبان‌های غیر انگلیسی، اگر بخواهیم تشخیص دهیم چه حرفی فشار داده شده است باید از KeyPress استفاده کنیم. در دو رویداد دیگر، حرف انگلیسی همان دکمه فشرده شده، قابل شناسایی است.

مثال۳ــ۲ : می‌خواهیم برنامه‌ای بنویسیم که با فشردن کلیدی، کاراکتر مربوطه را با استفاده از یک برچسب نمایش دهد (شکل ۱۲ــ۲).



شکل ۱۲ــ۲

الگوریتم یا روش انجام کار : با توجه به توضیحاتی که در مورد رویدادهای صفحه کلید داده شد، کافی است یک برچسب در روی فرم قرار دهیم و برای رویداد فشردن کلید در فرم یک متد EH بنویسیم که در آن کاراکتر کلید فشار داده شده را در درون برچسب نمایش دهیم. بدین منظور مراحل زیر را دنبال می‌کنیم :

۱ــ ایجاد پروژه جدید : وارد برنامه VS شوید و یک پروژه جدید از نوع WFA با نام KeyboardDemo در مسیر مشخصی بسازید.

۲ــ *تعیین ویژگی‌های فرم:* ویژگی‌های فرم را طبق جدول زیر تعیین کنید.

جدول ۹ــ۲ــ ویژگی‌های فرم

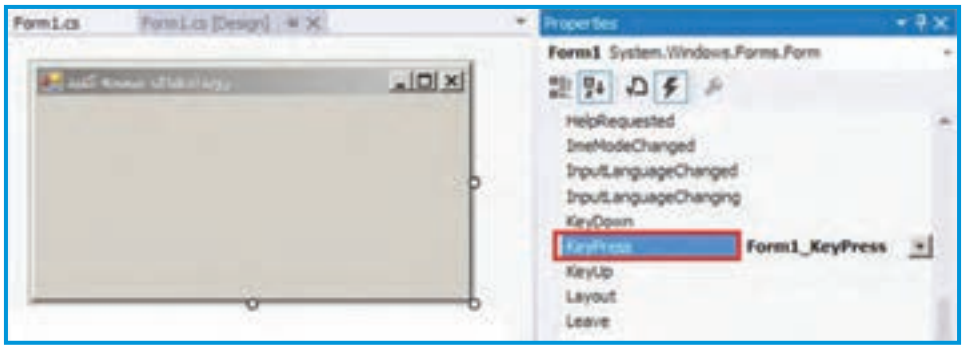
Form		
ویژگی		مقدار
Name		Form1
Text		رویدادهای صفحه کلید
Size	Width	300
	Height	200

۳ــ *تعیین ویژگی‌های برچسب:* یک برچسب به فرم اضافه کنید و ویژگی‌های آن را طبق جدول زیر تعیین کنید.

جدول ۱۰ــ۲ــ ویژگی‌های برچسب

Label		
ویژگی		مقدار
Name		Label1
Text		
AutoSize		True
RightToLeft		Yes
Font		Times New Roman Size: 14
Location	X	100
	Y	30

۴ــ ایجاد واکنش نسبت به صفحه کلید : مطابق شکل ۱۳ــ۲ فرم را انتخاب کرده و در پنجره Properties، رویداد KeyPress را پیدا کرده و دوبار کلیک کنید تا متد EH برای آن ایجاد شود.



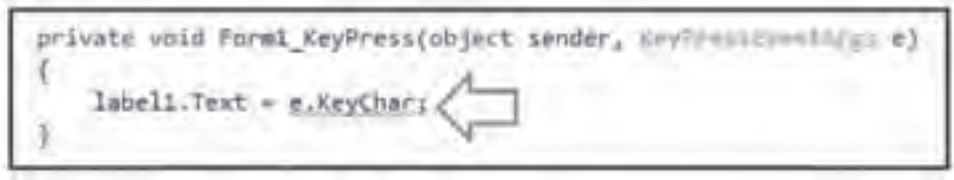
شکل ۱۳ ــ۲ــ ویژگی‌های فرم

متد EH رویداد KeyPress چنین خواهد بود :

```
private void Form1_KeyPress (object sender, KeyPressEventArgs e)
{
    label1.Text = "کلید زده شده :" + e.KeyChar;
}
```

با استفاده از پارامتر e که از نوع KeyPressEventArgs و ویژگی KeyChar، می‌توان کاراکتر متناظر با کلید زده شده را به‌دست آورد، که آن را از طریق برچسب نمایش می‌دهیم. اکنون برنامه را اجرا کنید و کلیدهای مختلفی را بزنید و نمایش آن را روی برچسب مشاهده کنید. کلیدهای غیرچاپی را نیز آزمایش کنید. آیا کاراکتری نشان داده می‌شود؟ آیا برنامه فعلی قادر به تشخیص آن می‌باشد.

سؤال: اگر در کد بالا برای نمایش کاراکتر وارد شده در برچسب، قبل از نمایش ویژگی e.KeyChar، پیام قرار داده نشود، خطا پدید می‌آید (شکل ۱۳ــ۲). چگونه می‌توان این خطا را رفع کرد؟



شکل ۱۴ ــ۲ــ خطای ناشی از مقداردهی نادرست ویژگی برچسب

سؤال: جهت نمایش کد کاراکتر وارد شده، به جای نمایش حرف آن چه تغییری باید در کد برنامه داده شود؟

توسعه و بهبود برنامه

برای اطلاع از فشردن کلیدهای غیر چاپی از رویداد KeyDown استفاده می‌کنیم :

مثال ۴ــ۲ : برنامه مثال قبل را توسعه دهید به طوری که قادر به تشخیص کلیدهای غیر چاپی نیز باشد.

مراحل زیر را برای انجام این کار دنبال می‌کنیم :

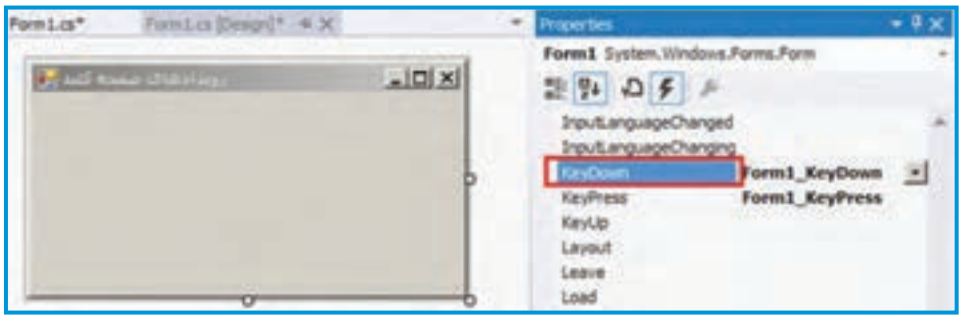
۱ــ پروژه مثال قبل که برای تشخیص کلید فشارداده شده بود را باز کنید.

۲ــ یک برچسب دیگر به فرم اضافه کنید. این برچسب برای اطلاع از اینکه کلید چاپی یا غیر چاپی فشار داده شده به‌کار می‌رود.

جدول ۱۱ــ۲ــ ویژگی‌های برچسب

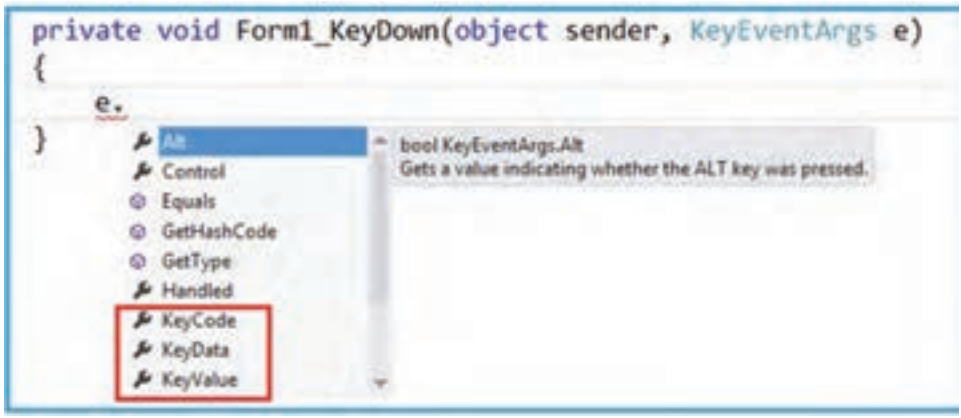
Label		
ویژگی		مقدار
Name		Label2
Text		
AutoSize		True
RightToLeft		Yes
Font		Times New Roman Size: 14
Location	X	**50**
	Y	80

۳ــ مطابق شکل ۱۴ــ۲ فرم را انتخاب کرده و در پنجره Properties، رویداد KeyDown را پیدا کرده و دوبار کلیک کنید تا متد EH برای آن ایجاد شود.



شکل ۱۴ــ۲

۴ــ متد EH رویداد KeyDown دارای پارامتری به نام e از نوع KeyEventArgs است که دارای ویژگی‌های مختلفی است که در شکل ۱۵ــ۲ به وسیلهٔ منوی IntelliSense تعدادی از ویژگی‌های مختلف آن لیست شده است. در جدول ۱۲ــ۲ کاربرد هر یک از ویژگی‌ها را مشاهده می‌کنید :



شکل ۱۵ ــ۲ــ ویژگی‌های پارامتر e

جدول ۱۲ــ۲ــ شرح ویژگی‌های پارامتر e

نام ویژگی	شرح ویژگی
Alt	اگر true باشد یعنی کلید Alt زده شده است.
Control	اگر true باشد یعنی کلید Ctrl زده شده است.
Shift	اگر true باشد یعنی کلید Shift زده شده است.
KeyCode	فقط نام کلید زده شده را اطلاع می‌دهد. اما قادر نیست ترکیب کلیدی زده شده را اطلاع دهد.
KeyData	نام کلید یا ترکیب کلیدی زده شده را اطلاع می‌دهد (مانند Ctrl + P ــ اگر بخواهیم بدانیم کلید همراه کدام کلید کنترلی بوده keydata مناسب است).
KeyValue	عدد صحیحی به عنوان کد کلید زده شده، تولید می‌کند.
SuppressKeyPress	یک متغیر Boolean است. دادن مقدار true به این ویژگی مانع از اجرا و تأثیر کلید روی کنترل می‌شود.

اکنون دستوری برای نمایش کلید یا ترکیب کلیدی زده شده درمتد EH رویداد KeyDown مطابق با شکل می‌نویسیم تا با مقدار خروجی و طریقه استفاده از ویژگی‌های مختلف جدول۱۲ــ۲ آشنا شویم.

```
private void Form1_KeyDown (object sender, KeyEventArgs e)
{
    label2.Text = "KeyData: " + e.KeyData+ "\n" +
                  "KeyCode: " + e.KeyCode+ "\n" +
                  "KeyValue: " + e.KeyValue+ "\n";
}
```

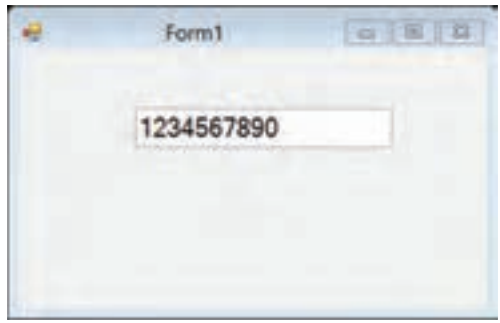
شکل ۱۶ــ۲

اکنون برنامه را اجرا کرده و کلیدهای مختلفی از هر دو گروه چاپی و غیرچاپی را بزنید و عکس‌العمل برنامه را مشاهده کنید. مثلاً اگر کلید shift را نگه داشته و حرف P را بزنید خروجی برنامه چنین خواهد شد :

در این حالت، عدد ۸۰ کد کاراکتر P است. به مقدار خروجی KeyData و KeyCode توجه کنید.

کار در کارگاه۳: ساخت یک کادر متنی عددی

می‌خواهیم کادر متنی ایجاد کنیم که فقط ارقام را بتوان در آن وارد کرد و همچنین کلیدهای Back space و Delete قابل اعمال باشند.



شکل ۱۷ــ۲

الگوریتم و روش انجام کار : برای این کار زمانی که کلیدی زده می‌شود باید بررسی شود اگر کلید زده شده غیر از عدد، Back space یا Delete باشد مانع از اجرای کلید زده شده شود.

۱ــ ***ایجاد یک پروژه جدید:*** در برنامه VS یک پروژه جدید از نوع WF A با نام NumeralTextBox در مسیر مشخصی بسازید.

۲ــ ***تعیین ویژگی‌های کادر متن:*** یک Text Box روی فرم قرار دهید و ویژگی‌های آن را به صورت جدول ۱۳ــ۲ تنظیم کنید.

جدول ۱۳ــ۲ــ ویژگی‌های کادر متن

TextBox		
ویژگی		مقدار
Name		input
Font	Size	14

۳ــ ایجاد واکنش کادر متنی نسبت به صفحه کلید: پس از انجام کارهای مربوط به طراحی واسط کاربری، نوبت کد نویسی است. برای آنکه از کلید زده شده با خبر بشویم و همچنین بتوانیم در صورت لزوم مانع از اجرای کلیدهای خاصی بشویم از رویداد KeyDown کنترل input استفاده می‌کنیم. در این رویداد، بررسی کلید زده شده صورت می‌گیرد. از طریق پنجره properties و قسمت رویدادها، رویداد keyDown را برای کنترل input ایجاد کنید و کدهای زیر را در آن بنویسید:

```
private void input_KeyDown (object sender, KeyEventArgs e)
{
    if (!(char.IsDigit( (Char)e.KeyCode) || e.KeyCode == Keys.Back ||
        e. KeyCode == Keys. Delete) )
        e. Suppresskeypress.true;
}
```

در کد بالا به نکات زیر توجه کنید:

۱ــ متد **IsDigit** برای ساختار char مشخص می‌کند کراکتر ورودی عدد است یا خیر؟ به عنوان نمونه جواب char.IsDigit('1') برابر true خواهد بود.

۲ــ توسط دستور (char)e.KeyCode تبدیل نوع کد کلید به حرف متناظر صورت می پذیرد.

۳ــ نوع شمارشی **Keys** حاوی مقادیر کدها با نام کلید مربوط و مقدار عددی کد آن کلید می‌باشد. به لحاظ شمارشی بودن و نمایش اسامی، کار برای مقایسه کلید زده شده با کلید مورد نظرمان راحت تر می‌شود. در کد بالا وقتی می‌خواهیم ببینیم آیا کلید زده شده Back space است یا خیر، از عبارت منطقی e.KeyCode == Keys.Back استفاده می‌کنیم زیرا Keys.Back نشان دهنده مقدار کلید Back space است. با توجه به همین موضوع، Keys.Delete نمایانگر چه کلیدی است؟

۴ــ با عملگر نقیض "!" که پشت کل عبارت منطقی شرط آمده است مشخص می‌کنیم اگر عبارت منطقی ما درست نباشد (کاراکتر وارد شده،حرف رقم، backspace یا Delete نباشد) مانع از اجرای کلید روی کنترل input شود. دستور **e.SuppressKeyPress=true** این کار را در برنامه انجام می‌دهد.

توسعه و بهبود برنامه

اگر کلیدهای جهت نما را بر روی کنترل کادر متنی امتحان کنیم خواهیم دید که عمل نمی‌کنند چرا؟ با استفاده از نوع شمارشی Keys و مقادیر Keys.Right و Keys.Left می‌توانیم کلیدهای جهت نما راست و چپ را با کلید زده شده مقایسه کنیم. کد رویداد را به صورت زیر اصلاح کنید.

```
private void input_KeyDown (object sender, KeyEventArgs e)
{
    if (!(char.IsDigit( (Char)e.KeyCode) || e.KeyCode == Keys.Back ||
        e. KeyCode = = Keys.Delete ||
        e. KeyCode = = Keys.Right || e.KeyCode == Keys.Left) )
            e.SuppressKeyPress = true;
}
```

سؤال: در نوع داده شمارشی Keys برای دکمه Enter چه مقداری در نظر گرفته شده است؟

خودآزمایی فصل دوم

الف) درستی یا نادرستی عبارات زیر را تعیین کنید :

۱ــ برای ایجاد واکنش نسبت به حرکت ماوس از رویداد MouseMove استفاده می‌شود.

۲ــ با استفاده از ویژگی Cursor در کنترل‌ها می‌توان شکل ماوس را تغییر داد.

۳ــ با استفاده از ویژگی e.KeyChar در رویداد KeyPress می‌توان فهمید کدام کلید فشار داده شده است.

۴ــ با استفاده از رویدادهای KeyDown و KeyUp نمی‌توان از فشردن کلیدهای غیرچاپی مطلع شد.

۵ــ اگر مقدار ویژگی SppressKeyPress یک کنترل False باشد، مانع از اجرا و تأثیرکلید روی کنترل می‌شود.

ب) جاهای خالی را با عبارت مناسب پر کنید :

۶ــ به وسیلهٔ پارامتر e.X و e.Y در متد MouseMove، می‌توانیم به ..... ماوس دسترسی داشته باشیم.

۷ــ برای دسترسی به کد کلید فشاردادهشده در رویداد keydown از ویژگی ..... پارامتر e استفاده می‌شود.

۸ــ اگر بخواهیم ببینیم چه حرفی فشار داده شده، باید از رویداد ..... استفاده کنیم.

ج) جدول زیر را تکمیل کنید.

	عملیات	رویداد پیشنهادی
۹	با ورود ماوس روی کادر متنی، رنگ کادر متنی تغییر کند.	
۱۰	با فشردن دکمه ماوس بر روی کنترل دکمه، متن کنترل تغییر کند.	
۱۱	با نگه‌داشتن ماوس بر روی کنترل قاب (GroupBox) پیامی نمایش داده شود.	
۱۲	با خارج‌شدن ماوس از برچسب، رنگ فرم تغییر کند.	

د) به سؤالات زیر پاسخ دهید.

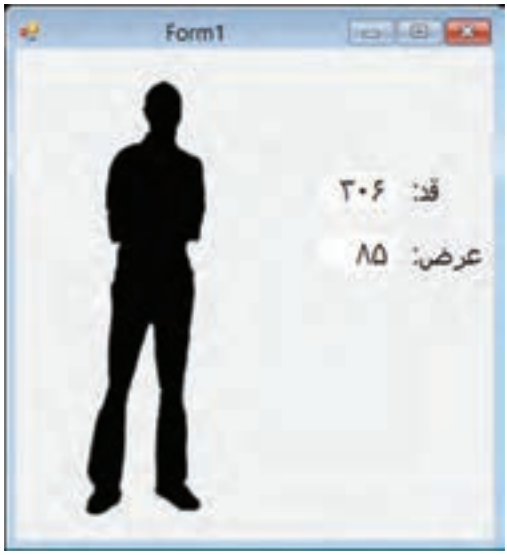
۱۳ــ دستوری بنویسید که در رویداد KeyPress کنترل کادر متنی، کد عددی کاراکتر وارد شده در کنترل برچسب نمایش داده شود.

۱۴ــ برای نمایش کد کلید shift از کدام رویداد صفحه کلید می‌توان استفاده کرد؟

تمرینات برنامه‌نویسی فصل دوم

۱ــ پروژه‌ای بنویسید که کد اسکی کاراکتر وارد شده را روی فرم نمایش دهد. این برنامه را برای کاراکترهای فارسی و کاراکترهای بزرگ و کوچک لاتین و اعداد اجرا نمایید.

۲ــ پروژه‌ای ایجاد کنید که قد و عرض تصویر شخص را با دکمه‌های جهت‌نما تغییر دهد. همچنین قد و عرض را (براساس پیکسل) در Labelهای مربوط درج کند. کارکرد دکمه‌های جهت‌نما به این صورت است :



بالا : افزایش قد (+۱۰)

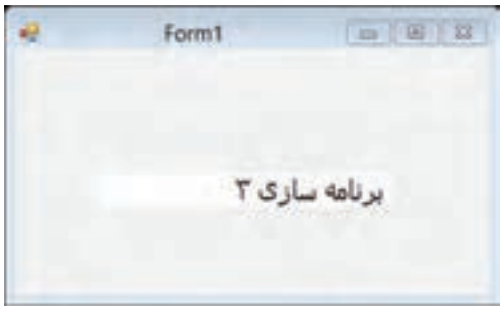
پایین : کاهش قد (۱۰–)

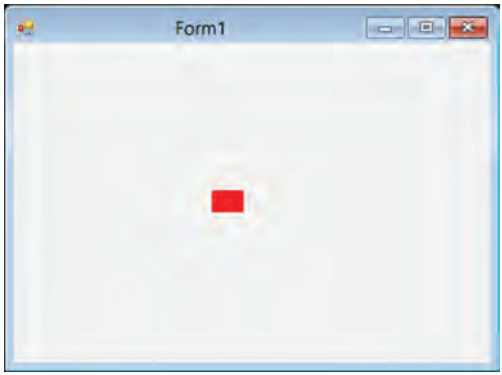
راست : افزایش عرض (۱۰ +)

چپ : کاهش عرض (۱۰ –)

۳ــ فرم ساده‌ای بسازید که با گرفتن یک قسمت داخلی آن (غیر از نوار عنوان)، و کشیدن به یک محل دیگر از صفحه، فرم جابه‌جا شود. درست همان‌طوری که با کشیدن نوار عنوان، این کار انجام می‌شود (از ویژگی‌های left و Top فرم استفاده کنید).

۴ــ فرمی بسازید که دارای یک برچسب باشد که تا حدودی نقش کادر متنی را بازی کند. به این معنا که هر حرفی، به وسیلهٔ صفحه، کلید زده شود به متن برچسب اضافه شود. همچنین اگر Back Space زده شود حرف آخر پاک شود.





۵ــ یک بازی دو نفره طراحی کنید که نفر اول، با دابل کلیک اشاره‌گر ماوس را پنهان کند. آنگاه بازیکن دوم تا ۵ بار فرصت داشته باشد روی محل هدف کلیک کند. درصورت موفقیت، پیام پیروزی ظاهر شود و با کلیک روی هدف یا سپری شدن ۵ بار فرصت کلیک، اشاره گر ماوس ظاهر شود.

برای پنهان شدن اشاره‌گر ماوس از دستور Cursor.Hide() و برای ظاهر شدن آن از دستور Cursor.Show() استفاده کنید.

۶ــ پروژه‌ای بسازید که به‌عنوان یک بازی به صورت زیر استفاده شود. از سمت راست و بالا یک سفینه به سمت چپ حرکت کرده و به تدریج به پایین نزدیک می‌شود. اگر به زمین برخورد کند یک امتیاز کم می‌شود. بازیکن باید به محض دیدن سفینه روی آن کلیک کند تا به این ترتیب سفینه به ارتفاع بالاتری صعود کند تا به تدریج از سمت چپ خارج شود. اگر سفینه از سمت چپ خارج شود یک امتیاز برای بازیکن محسوب می شود. تعداد سفینه‌ها پنج عدد است.

فعّالیت

تکمیل پروژه

در فصل قبل پروژه واژه نامه را ایجاد کردیم و دکمه جستجو را کدنویسی نمودیم. در این فصل رویدادهای ماوس مربوط به این پروژه را تکمیل می‌کنیم.

مرحله دوم:

۱ــ کاربر ممکن است برای جستجو از حروفی استفاده کند که در یک واژه به کار برده نمی‌شود. بنابراین می‌خواهیم اگر از حروف غیر معمول استفاده شود پیام مناسبی ارایه شود.

۲ــ همچنین می‌خواهیم با ورود اشاره‌گر ماوس به قسمت «معنی و توضیح» پس زمینه آن سفید رنگ شود و با خروج اشاره‌گر ماوس، به حالت قبلی برگردد.

ورود اشاره‌گر ماوس

خروج ماوس

۳ــ می‌خواهیم جستجو علاوه بر زدن دکمه، با زدن کلید Enter در کادر واژه نیز انجام شود.

نحوه انجام:

الف) متد EH رویداد KeyPress برای کادر متنی searchWord را این‌گونه معین کنید :

```
private void searchword_KeyPress (object sender, KeyPressEventArgs e)
{
    if (!(char.IsLetter ( e.KeyChar) || e.KeyChar ==' ' || e. KeyChar =='\b' ) )
    {
        MessageBox.Show( "حرف نامعتبری برای واژه وارد شده است");
    }
}
```

شرطی که در کد آمده است با عملگر نقیضی که قبل از پرانتز درونی آمده، بررسی می‌کند اگر هیچ‌کدام از عبارت‌های منطقی زیر صحیح نباشد پیام را نمایش دهد :

- حرف وارد شده جزء حروف الفبایی باشد.
- حرف وارد شده برابر با فاصله باشد.
- حرف وارد شده برابر با Back Space باشد.

نکته

متد IsLetter برای ساختار char بررسی می‌کند که ورودی آن حرف الفبایی است یا خیر.

ب) برای اینکه رنگ پس زمینه با رفتن ماوس روی کادر معنی سفید شود و با خارج شدنش به رنگ قبلی بازگردد، لازم است بتوانیم مقدار رنگ را در متغیری ذخیره کنیم. به نظر شما این متغیر از چه نوعی باید باشد؟ به زبان فارسی نوع «رنگ» و به زبان برنامه نویسی Color است.

بنابراین می‌توان متغیری به صورت زیر تعریف کرد :

```
Color  Bcolor;
```

این متغیر را به صورت سراسری در کلاس فرم تعریف می‌نماییم تا در متدهای مختلف بتوانیم از آن استفاده کنیم و مقدار خود را نیز در طول اجرای برنامه حفظ کند.

ج) متد EH رویداد MouseEnter برای کنترل description را به صورت زیر بنویسید.

```
private void description_MouseEnter(object sender, EventArgs e)
{
    Bcolor = description.BackColor;
    description.BackColor = Color.White;
}
```

همان‌طور که مشخص است در خط اول کد درون متد، رنگ فعلی پس زمینه کادر description به متغیر Bcolor داده می‌شود. با این عمل مقدار رنگ فعلی، برای استفاده بعدی نگهداری می‌شود.

در خط دوم رنگ سفید به مقدار پس زمینه کادر داده می‌شود.

همان‌طور که دیده می‌شود ساختار Color دارای مقادیر مختلفی از رنگ‌ها است. اگر بررسی بیشتری داشته باشید این ساختار، متدی مانند FromArgb برای مقداردهی دقیق رنگ دارد که توسط مقدار چهار مؤلفه یعنی سه رنگ قرمز،سبز، آبی و همچنین مؤلفه میزان شفافیت، رنگ موردنظر را ایجاد می‌کند.

د) متد EH رویداد MouseLeave برای کنترل description را به صورت زیر بنویسید.

```
private void description_MouseLeave(object sender, EventArgs e)
{
    description.BackColor = Bcolor;
}
```

توسط کد این متد، رنگ ابتدایی که در متغیر Bcolor نگهداری شده است را به رنگ پس‌زمینه کادر متنی description نسبت می‌دهیم.

ه) برای اینکه عمل جستجو با زدن دکمه Enter انجام شود کد زیر را می‌نویسیم.

```
private void searchword_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode ==Keys.Enter)
        search.PerformClick( );
}
```

نکته

دستور search.PerformClick() باعث اجرای رویداد کلیک در دکمه search می‌شود.

عمل شماره ۳ همچنین با ویژگی AcceptButton فرم قابل انجام است. در این صورت، دکمه search را برای این ویژگی تنظیم می‌کنیم و نیاز به نوشتن کد ندارد.

واژگان و اصطلاحات انگلیسی فصل دوم

ردیف	لغت انگلیسی	معنی لغت به فارسی
۱	Arrow Key	
۲	Character	
۳	Coordinate	
۴	Cross	
۵	Cursor	
۶	Default	
۷	Demo	
۸	Down	
۹	Event	
۱۰	Height	
۱۱	Hover	
۱۲	Interactive	
۱۳	Leave	
۱۴	Location	
۱۵	Pixel	
۱۶	Play	
۱۷	Point	
۱۸	Press	
۱۹	printable	
۲۰	Tunnel	
۲۱	Up	
۲۲	Victory	
۲۳	Width	

فصل سوم

منو

یکی از وظایف طراحان نرم افزار این است که سیستم‌ها را طوری طراحی کنند که بهره‌برداری از آنها آسان باشد. داشتن منوهای مناسب و سپردن حداقل کار به کاربر، از جمله مسائلی است که به این امر کمک می‌کند. منوها جزء تفکیک ناپذیر هر برنامهٔ خوب محسوب می‌شوند و راهی راحت و پرکاربرد برای دسترسی کاربر به تمام قسمت‌های برنامه را فراهم می‌کنند. برای مثال برنامهٔ ویژوال استودیو با استفاده از منوها، به برنامه نویس این امکان را می‌دهد که به راحتی به ابزارهای این محیط دسترسی داشته باشد و بتواند از آنها استفاده کند. در این فصل می‌خواهیم روش ایجاد منوی مناسب برای برنامه را یاد بگیریم.

پس از پایان این فصل انتظار می‌رود که فراگیر بتواند :

۱ــ دلایل استفاده از منو را بیان کند و منوی مناسبی برای برنامه خود تعریف نماید.

۲ــ منوهای نواری و زمینه را در واسط گرافیکی کاربر پسند مورد استفاده قرار دهد.

۳ــ تنظیمات ویژه‌ای روی منو و گزینه‌های آن انجام دهد.

۱ــ۳ــ استفاده از منو[1] در واسط گرافیکی کاربر

اغلب برنامه‌های کاربردی مجهز به منو می‌باشند که باعث دسته بندی موضوعی و امکانات یک برنامه می‌شود. با استفاده از منو، کاربر راحت تر می‌تواند به آنچه که مورد نظرش است، برسد. معمولاً گزینه‌های منو به دو روش قابل دسترسی هستند :

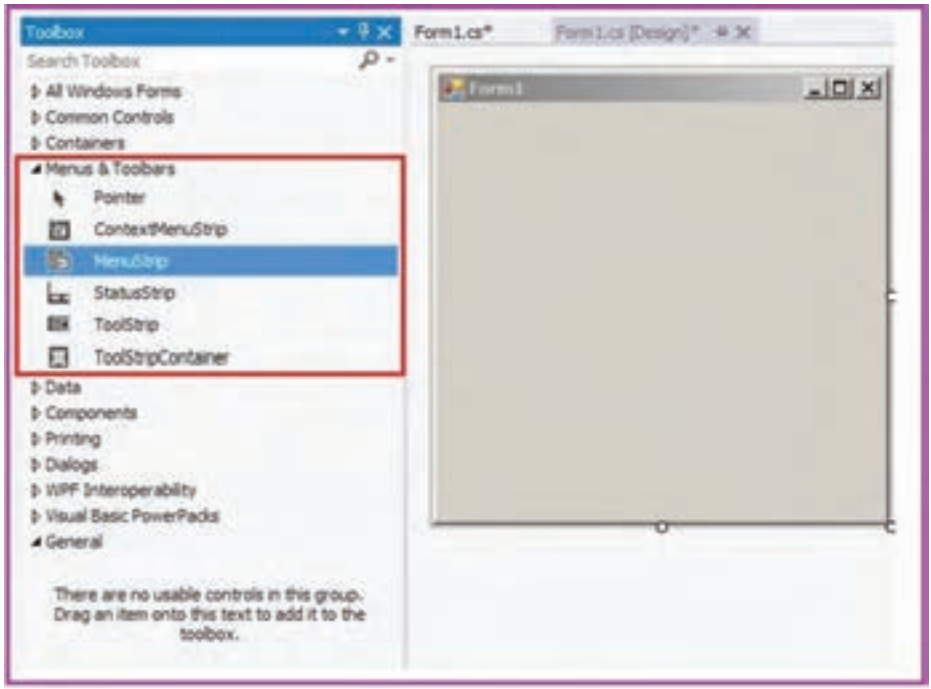
۱ــ وارد کردن یک کلید خاص مانند کلیدهای دسترسی[2] و یا میانبر[3]

۲ــ انتقال مکان‌نما به گزینه موردنظر و فشردن کلید Enter یا کلیک ماوس

اکنون برای اضافه‌کردن منو به برنامه، مراحل زیر را دنبال کنید :

۱ــ برای ایجاد منو، یک پروژه جدید از نوع ویندوز بسازید.

۲ــ در محیط VS، در حالی که پنجره طراحی فرم دیده می‌شود سراغ پنجره جعبه ابزار رفته و در جعبه ابزار فرعی Menu & Tool Bars ابزار MenuStrip را پیدا کنید و آن را به فرم اضافه نمایید (شکل ۱ــ۳).



شکل ۱ــ۳ــ نحوهٔ دسترسی به ابزار منو

۱ــ Menu

۲ــ Hot Key

۳ــ Shortcut Key

ازنظر ظاهری با اضافه‌کردن ابزار منو (MenuStrip) به فرم، دو اتفاق می‌افتد :

۱ــ یک شیء به نام menuStrip1 در زیر فرم قرار می‌گیرد.

۲ــ در زیر خط عنوان پنجره فرم، نوار منویی قرار می‌گیرد که در داخل آن می‌توانید گزینه‌های منوهای افقی، عمودی و زیر منوهای مورد نظر خود را تایپ کنید (شکل ۲ــ۳).

شکل ۲ــ۳ــ نحوهٔ کار با شیء منو

۳ــ در اولین عنوان، عبارت Type Here ظاهر می‌شود، برای تغییر نام، روی این عنوان کلیک کرده، متن جدید را تایپ و کلید Enter را فشار دهید. با استفاده از کلیدهای مکان نما یا ماوس می‌توان بین گزینه‌های منو حرکت کرده، گزینه مورد نظر را انتخاب کرد و عنوان آن را تغییر داد.

۴ــ زمانی که بر روی شیء menuStrip1 کلیک می‌کنیم، پنجره properties، ویژگی‌های آن منو را نمایش می‌دهد و مانند هر کنترل دیگری می‌توانید ویژگی‌های آن را تنظیم کنید (جدول۱ــ۳).

جدول ۱ــ۳ــ ویژگی‌های متداول شیء منو

ویژگی	شرح
Cheked	تیک دار کردن گزینه منو
Enabled	فعال یا غیرفعال کردن گزینه منو
Image	قرار دادن تصویر برای گزینه منو
Name	نام مربوط به منو یا گزینه‌های منو در کد نویسی
RightToLeft	راست به چپ کردن گزینه‌های منو (مانند زبان فارسی)
ShortCutKeys	تعریف کلید میانبر
Text	گزینه‌های منو
Visible	نمایش گزینه‌های منو

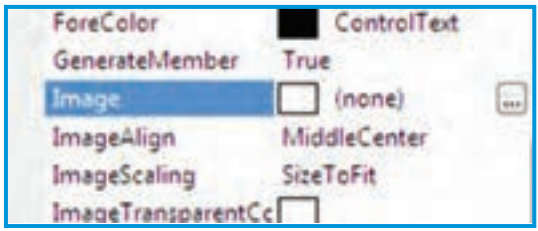
۲ــ۳ــ امکانات منو

برای منوهایی که به وسیله این کنترل ایجاد می‌کنید علاوه بر عنوانی که می‌توانید برای هر منو و گزینه‌های منو نمایش دهید، می‌توانید امکاناتی شامل آیکن (تصویر)، کلید دسترسی، کلید میان‌بر و یا حتی گزینه‌هایی با امکان قابلیت انتخاب را به‌کار گیرید (شکل۳ــ۳).

شکل ۳ــ۳ــ امکانات منو

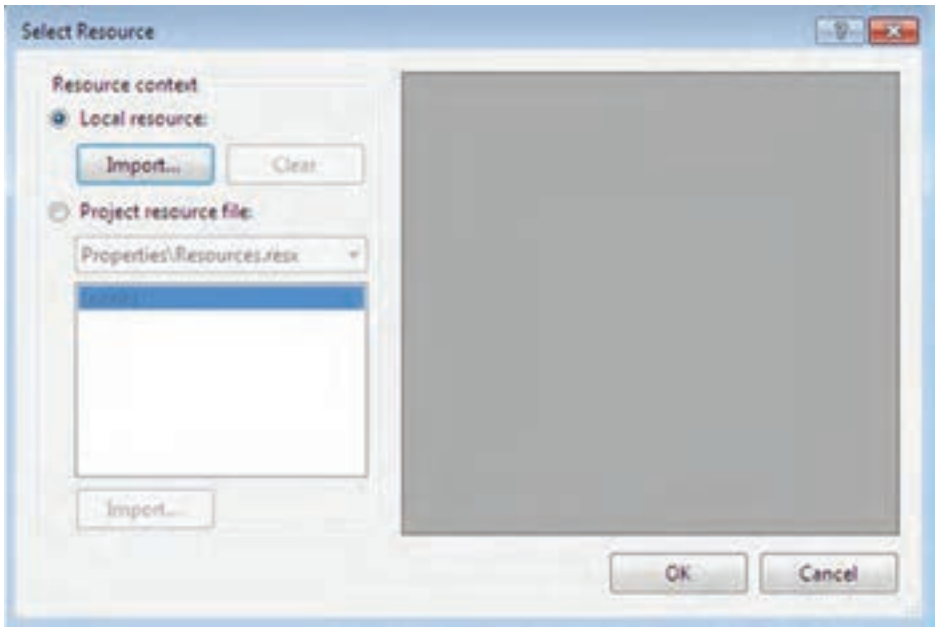
۱ــ۲ــ۳ــ جدا کننده بین گزینه‌ها : جداکننده[۱] به خط افقی گفته می‌شود که با قرار گرفتن در بین گزینه‌های منو، آنها را دسته‌بندی می‌کند. برای اضافه‌کردن جداکننده به گزینه‌های منو، باید در هنگام طراحی منو، در قسمت نام گزینه منو علامت – (منها) قرار داد.

۲ــ۲ــ۳ــ اضافه کردن آیکن به منوها : شما می‌توانید برای هر گزینه منو، یک تصویر کوچک را انتخاب کنید. که در سمت چپ آن نمایش داده شود برای این کار از خاصیت Image استفاده می‌شود. گزینه مورد نظر در منو را انتخاب نمایید. بر روی دکمه کوچک کنار خاصیت Image در پنجره Properties کلیک کنید.



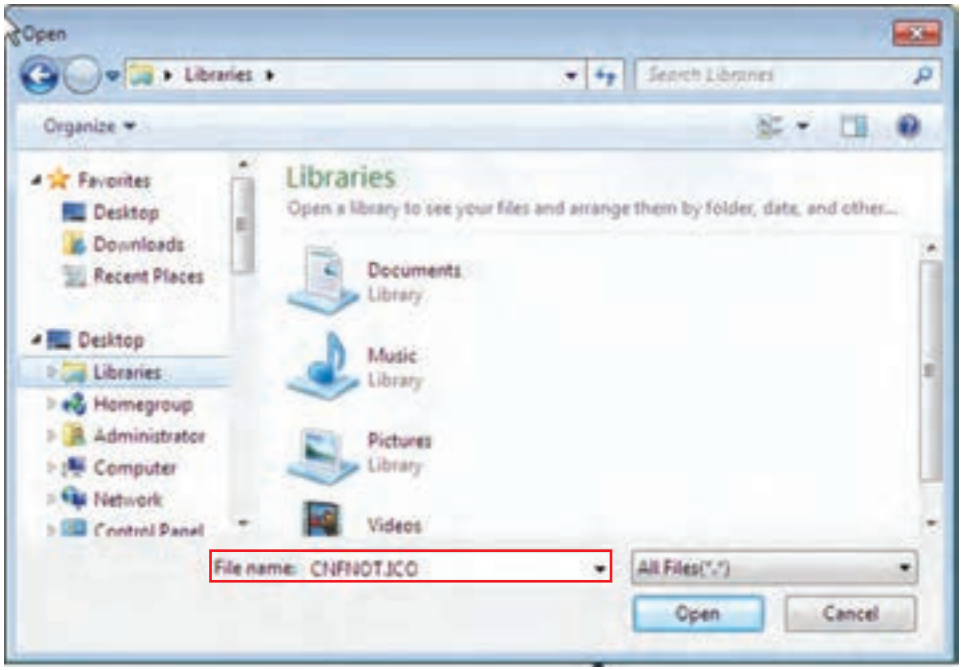
شکل ۴ــ۳ــ تعیین ویژگی Image گزینه منو

منبعی که عکس در آن قرار دارد را انتخاب و سپس مسیر آن‌را مشخص نمایید. اگر اندازه عکس بزرگ باشد، به صورت خودکار بر اساس مقدار خاصیت ImageScaling اندازه آن تغییر می‌کند.



شکل ۵ ــ۳ــ اضافه کردن تصویر به منابع پروژه با استفاده از دکمه Import

۱ــ Separator



شکل ۶ ــ۳ــ انتخاب تصویر از نوع ico. برای آیکن منو

نکته

برای آیکن (تصویر گزینه) هم از فایل‌های آیکن (ico.) و هم از تصاویر دلخواه (با فرمت‌های wmf.,gif.,bmp.,png.,jpg.) می‌توان استفاده کرد.

۳ــ۲ــ۳ــ ایجاد کلید دسترسی : کلیدهای دسترسی، آن دسته از کلیدها هستند که برای فعّال کردن منو و گزینه‌های آن با استفاده از صفحه کلید تعریف می‌شوند. که با فشردن هم‌زمان Alt و یک حرف خاص منجر به بازشدن منو می‌شوند.

برای ایجاد کلید دسترسی، قبل از کاراکتر مورد نظر از علامت & استفاده می‌کنیم.

&Edit ← نمایش ← Edit

در مثال بالا منوی Edit با استفاده از کلیدهای Alt+E قابل دسترسی است.

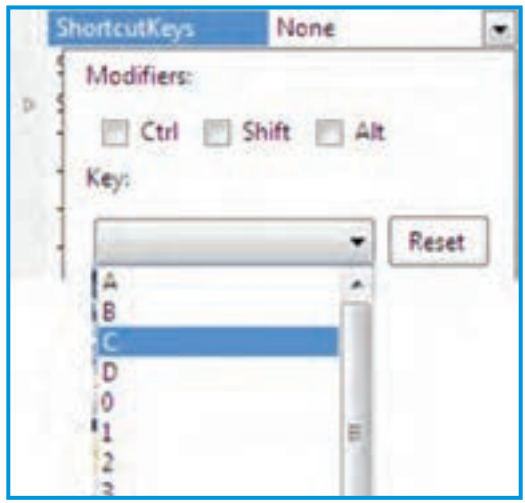
نکته

در زمان اجرای برنامه با فشردن کلید Alt، می‌توان کلیدهای دسترسی منوی تعریف شده را مشاهده کرد.

سؤال: آیا می‌توان از حروف فارسی به عنوان کلید دسترسی استفاده نمود؟

۴ــ۲ــ۳ــ ایجاد کلید میان‌بر : با استفاده از ویژگی ShortCutKeys، می‌توان کلید میان‌بر به منو اضافه کرد که باعث می‌شوند تا کاربران بتوانند عملکرد گزینه‌های منو را با سرعت بیشتری انجام دهند.

در پنجره properties در قسمت ShortcutKeys می‌توان کلید ترکیبی مورد نظر را انتخاب نمود. کلیدهای ترکیبی از ترکیب کلیدهای Ctrl یا Shift یا Alt به همراه حروف الفبا یا ارقام درست می‌شوند (شکل ۷ــ۳).



شکل ۷ــ۳ــ نحوه ایجاد کلید میان‌بر

سؤال: در استفاده از کلیدهای میان‌بر آیا بین حروف کوچک و بزرگ تفاوت وجود دارد؟

نکته

در انتخاب کلید ترکیبی برای کلید میان‌بر دقت کنید تا با کلیدهای میان‌بر ویندوز تداخل نداشته باشد.

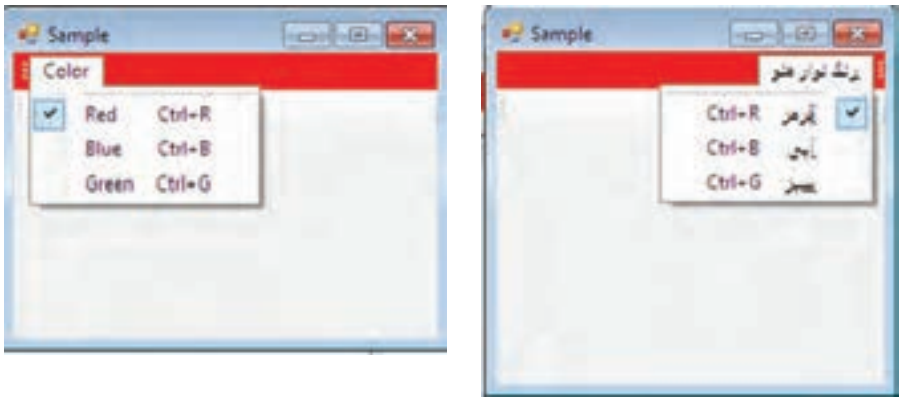
سؤال: در صورت انتخاب کلید میان‌بر مشترک با ویندوز، کدام دستور اجرا می‌شود ؟

کار در کارگاه ۱

مثال ۱ــ۳ : فرم‌طراحی کنید که با استفاده از گزینه‌های منو، رنگ نوار منو تغییر کند.

۱ــ درج کنترل منو : یک شیء از کنترل MenuStrip برای برنامه ایجاد می‌کنیم.

۲ــ تعیین نام و گزینه‌های منو : یک منو با نام Color ایجاد کرده و سپس برای آن چند گزینه رنگ (مثلاً Red , Blue , Green) در نظر می‌گیریم (شکل ۸ــ۳).



شکل ۸ــ۳ــ اجرای منوی طراحی شده برای مثال ۱ــ۱

۳ــ تعیین ویژگی منو : برای گزینه‌ها، کلید دسترسی و کلید میان‌بر تعریف می‌کنیم. برای علامت‌دار شدن گزینه مورد نظر از ویژگی Checked کمک می‌گیریم. می‌توان منوی فارسی ایجاد کرد. برای جهت آن از راست به چپ مقدار ویژگی Right TO Left را Yes در نظر می‌گیریم.

جدول ۲ــ۳ــ ویژگی‌های منو

ویژگی	مقدار		
name	RedToolstripMenuItem	blueToolstripMenuItem	greenToolstripMenuItem
Checked	True	false	false
ShortcutKey	Ctrl+R	Ctrl+B	Ctrl+G
Text	Red	Blue	Green

۳ــ۳ــ طراحی منوی استاندارد

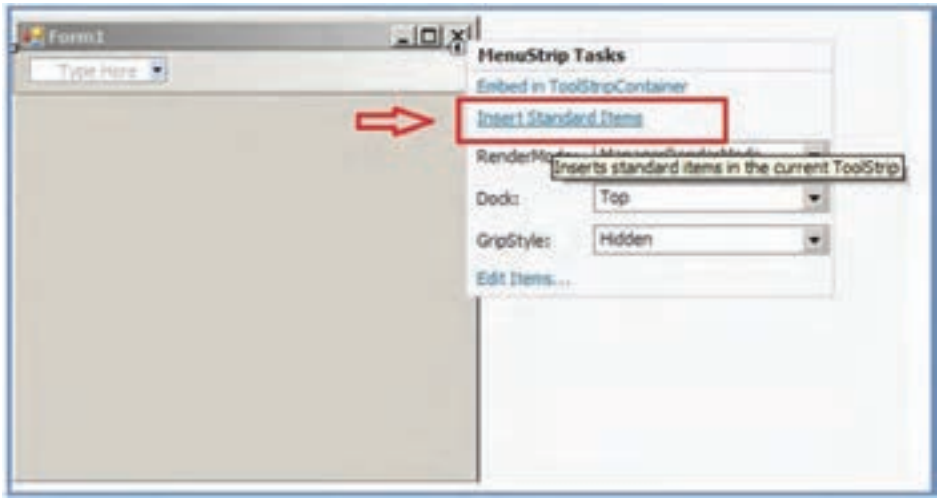
علاوه براینکه می‌توانید منوی دلخواه خود را بسازید، اگر مایل باشید می‌توانید منوی‌های استانداردی که معمولاً در برنامه‌های کاربردی ویندوز دیده می شود، به برنامه خود اضافه کنید.

مثال ۲ــ۳ : چگونه منوهای استاندارد به فرم خود اضافه کنیم؟

الگوریتم یا روش انجام کار :

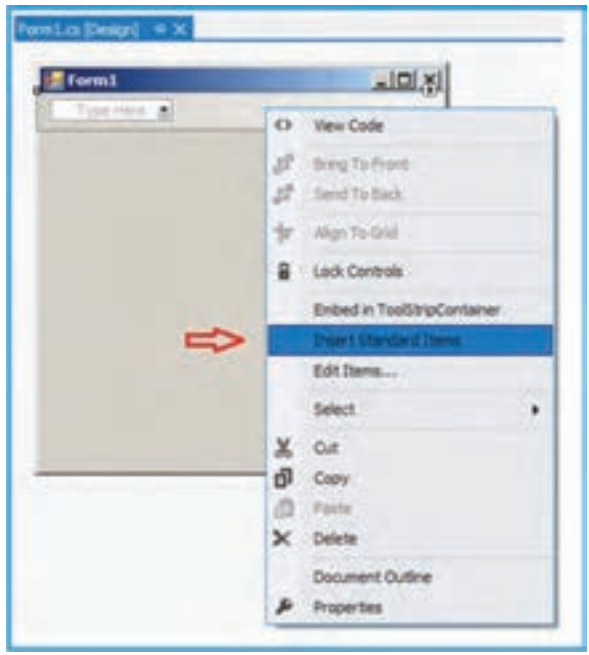
درج کنترل منو : ابتدا کنترل MenuStrip را به فرم اضافه می کنیم.

فراخوانی منوی استاندارد : کنترل منو را انتخاب و سپس روی مثلث مشکی در کادر آن کلیک کرده، لیستی ظاهر می‌شود. گزینه Insert Standard Items را انتخاب می کنیم (شکل ۹ــ۳).



شکل ۹ــ۳ــ نحوه انتخاب منوی استاندارد

همچنین می‌توانید به روش دیگر، بر روی نوار منوی MenuStrip کلیک راست کرده و از لیستی که ظاهر می‌شود، گزینه Insert Standard Items را انتخاب کنید (شکل ۱۰ــ۳).

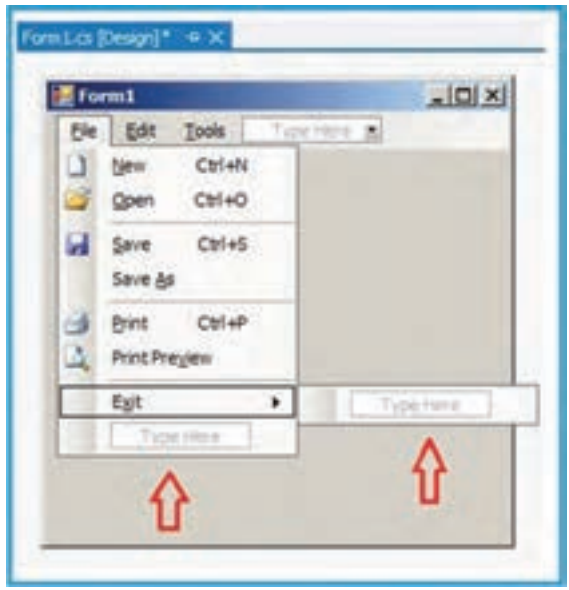


شکل ۱۰ــ۳ــ نحوه انتخاب منوی استاندارد با استفاده از کلیک راست روی نوار منو

در هریک از دو روش، منوی استاندارد به فرم مانند شکل ۱۱ــ۳ اضافه می‌شود.

شکل ۱۱ــ۳ــ نمایی از منوی استاندارد

پس از انجام عمل فوق، در بالای فرم، منوهای File، Edit، Tools، Help مشاهده می‌شود که در هر یک از آنها، گزینه‌های فرعی وجود دارند. در صورت نیاز می‌توانید این منو را بر طبق نیاز، تغییر دهید و گزینه‌های جدیدی را اضافه و یا حذف کنید (شکل ۱۲ــ۳).



شکل ۱۲ــ۳ــ گزینه منوی File از منوی استاندارد

با اجرای برنامه، منوهای استاندارد به همراه گزینه‌های فرعی آنها، دیده می‌شود اما با کلیک برروی هر یک از گزینه‌ها، اتفاقی نمی‌افتد. چرا؟

۴ــ۳ــ واکنش نسبت به گزینه‌های منو

برای فعال‌سازی گزینه‌های منو باید کدهای مربوط به آنها را نوشت. برای نوشتن کد مربوط به هر گزینه، روی آن دوبار کلیک کرده تا وارد محیط کدنویسی شده، دستور (دستورات) مورد نظر را بنویسیم. از متد EH برای این کار استفاده می‌کنیم. با توجه به اینکه عملیات ثبت رویداد و برقرارکردن ارتباط بین رویداد کلیک و متد EH به وسیلهٔ VS به طور خودکار انجام می‌شود. بنابراین کار دیگری به جز اجرای برنامه باقی نمی‌ماند.

مثال ۳ــ۳ : گزینه Exit در منوی File را برای خاتمه دادن به برنامه کدنویسی کنید.

الگوریتم یا روش انجام کار : باید یک متد EH برای گزینه Exit بنویسیم. بنابراین در پنجره طراحی فرم، با دوبار کلیک بر روی گزینه Exit، پنجره کد نویسی باز می‌شود در داخل متدی که برای این منظور توسط VS ایجاد شده است، دستور بستن پنجره را می‌نویسیم :

```
public partial class Form1 : Form
{
    public Form1( )
    {
        InitializeComponent ( );
    }
    private void exitToolStripMenuItem_Click(object sender,
    EventArgs e)
    {
        this.Close( );   ⇦ دستور بستن پنجره
    }
}
```

مثال ۴ــ۳ : می‌خواهیم برای مثال ۱ــ۲ که طراحی کردیم کدنویسی کنیم به‌طوری‌که با انتخاب رنگ مورد نظر، رنگ نوار منو تغییر کند و گزینه مورد نظر علامت‌دار شود.

الگوریتم یا روش انجام کار :

۱ــ نوشتن رویداد Click برای گزینه Red : روی گزینه "Red" دو بار کلیک کرده و کد زیر را وارد می‌کنیم :

```
private void redToolStripMenuItem_Click(object sender, EventArgs e)
{
    menuStripl. BackColor = Color. Red;   ⇦ تعیین رنگ نوار منو
    colorStripMenuItem.BackColor = Color.Red;   ⇦ تعیین رنگ گزینه منو
```

```
        redToolStripMenuItem.Checked = true;
        blueToolStripMenuItem.Checked = false;
        greenToolStripMenuItem.Checked = false;
    }
```

۲ــ نوشتن رویداد Click برای بقیه گزینه‌ها : برای گزینه‌های دیگر نیز مطابق با کد بالا می‌توان کدهای مربوط به رنگ "Green" و "Blue" را نیز نوشت.

مثال ۵ ــ۳ : پروژه‌ای ایجاد کنید که دارای سه منوی File و Edit و View باشد. دو کادر متنی روی فرم در نظر بگیرید. منوی File دارای گزینه New برای خالی کردن کادرهای متنی و گزینه Exit برای خروج از برنامه است. منوی Edit دارای دو گزینه Copy و Paste و منوی View دارای گزینه BackColor و گزینه ForeColor برای تعیین رنگ زمینه و نوشته‌های کادر متنی با استفاده از کادر محاوره‌ای رنگ است.

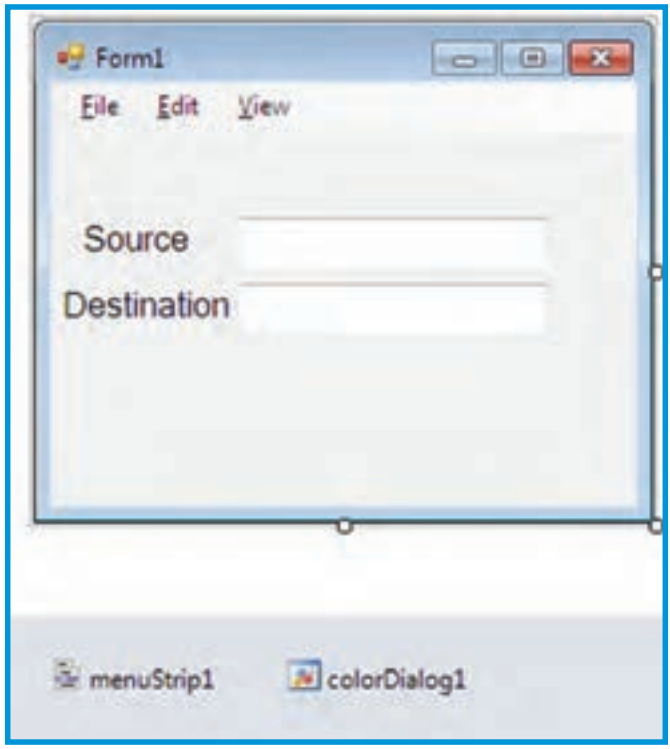
الگوریتم یا روش انجام کار :

۱ــ درج کنترل منو : با استفاده از ابزار Menu & Tool Bars یک شیء MenuStrip برای فرم ایجاد می‌کنیم و منوی استاندارد را برای آن فرا می‌خوانیم.

۲ــ ایجاد گزینه‌های مورد نظر برای منو : در منوی File ، گزینه‌های New و Exit را نگه داشته و بقیه گزینه‌ها را حذف می‌کنیم. در منوی Edit، به غیر از گزینه‌های Copy و Paste بقیه گزینه‌ها را حذف می‌نماییم. منوی Tools را به View تغییر نام داده و گزینه‌های قبلی را حذف کرده و دو گزینه BackColor و ForeColor را اضافه می‌نماییم. منوی Help را نیز حذف می‌کنیم.

۳ــ درج کنترل‌های کادر متنی : دو کادر متنی روی فرم قرار می‌دهیم.

۴ــ درج کنترل ColorDialog : از قسمت ابزار Dialogs کادر محاوره‌ای ColorDialog را برای فرم در نظر می‌گیریم (شکل ۱۳ــ۳ و شکل ۱۴ــ۳).



شکل ۱۴ــ۳ــ فرم مثال ۵ــ۳

شکل ۱۳ــ۳ــ جعبه ابزار Dialogs

۵ــ نوشتن رویداد Click برای گزینه New : روی گزینه New دو بار کلیک کرده و کدهای زیر را می‌نویسیم :

```
textBox1.Text = " ";
textBox2.Text = " ";
```

۶ــ نوشتن رویداد Click برای گزینه Exit : روی گزینه Exit دو بار کلیک کرده و کد زیر را می‌نویسیم :

```
this.Close();
```

۷ــ متغیر سراسری s1 را در داخل کلاس Form1 تعریف کنید :

```
string s1;
```

۸ــ نوشتن رویداد Click برای گزینه Copy : روی گزینه Copy دو بار کلیک کرده و کد زیر را می‌نویسیم :

```
S1 = textBox1.SelectedText;
```

۹ــ نوشتن رویداد Click برای گزینه Paste : روی گزینه Paste دو بار کلیک کرده و کد زیر را می‌نویسیم :

```
textBox2.Text = s1;
```

سؤال: کدهای گزینه Copy را طوری تغییر دهید که اگر متنی انتخاب نشده پیغام خطا بدهد.

۱۰ــ نوشتن رویداد Click برای گزینه BackColor : روی گزینه BackColor دو بار کلیک کرده و کدهای زیر را می‌نویسیم :

```
colorDialog1.ShowDialog( );
textBox1.BackColor = colorDialog1.Color;
textBox2.BackColor = colorDialog1.Color;
```

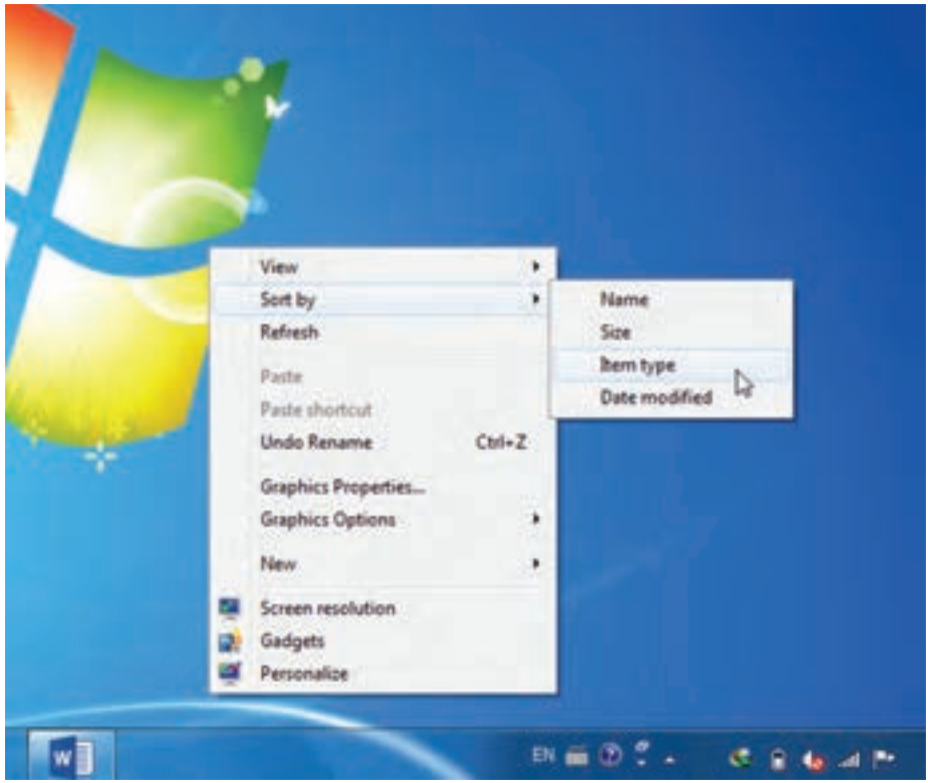
۱۱ــ نوشتن رویداد Click برای گزینه ForeColor : روی گزینه ForeColor دو بار کلیک کرده و کدهای زیر را می‌نویسیم :

```
colorDialog1.ShowDialog();
textBox1.BackColor = colorDialog1.Color;
textBox2.ForeColor = colorDialog1.Color;
```

سؤال: چرا بهتر است برای تغییر گزینه منو به جای تغییر نام، آن گزینه را حذف کرده و گزینه جدید تعریف شود؟

۵ــ۳ــ منوی زمینه[۱]

شکل ۱۵ــ۳ را بارها روی میز کار ویندوز دیده‌اید. این‌گونه منوها که با کلیک راست بر روی زمینه پدید می‌آیند، منوی زمینه یا میان‌بر نام دارند. این منوها در هر موقعیت مکانی از صفحه، ممکن است دارای گزینه‌های متفاوتی باشند.



شکل ۱۵ــ۳ــ منوی زمینه با کلیک راست روی دسک تاپ ویندوز

کنترل منوی زمینه در گروه Menu & Tool Bars از جعبه ابزار (ToolBox) قرار دارد. نام این کنترل ContextMenuStrip است. نحوه استفاده از این کنترل را در مثال صفحهٔ بعد ببینید.

۱ــ Context Menu

مثال ۶ــ۳ : می‌خواهیم یک ویرایشگر متنی داشته باشیم که فرمان تغییر نوع قلم، رنگ متن و رنگ پس زمینه آن از طریق منوی کلیک راست قابل دسترسی باشد (شکل ۱۶ــ۳).

شکل ۱۶ــ۳ــ ویرایشگر متنی با منوی زمینه

الگوریتم یا روش انجام کار :

۱ــ ایجاد پروژه : یک پروژه جدید از نوع WFA با قراردادن یک کادر متنی در فرم ایجاد نمایید.

۲ــ تنظیم ویژگی‌های کادر متنی

جدول ۳ــ۳ــ ویژگی‌های کادر متنی

Text Box	
ویژگی	مقدار
Name	editor
Dock	fill
Multiline	True

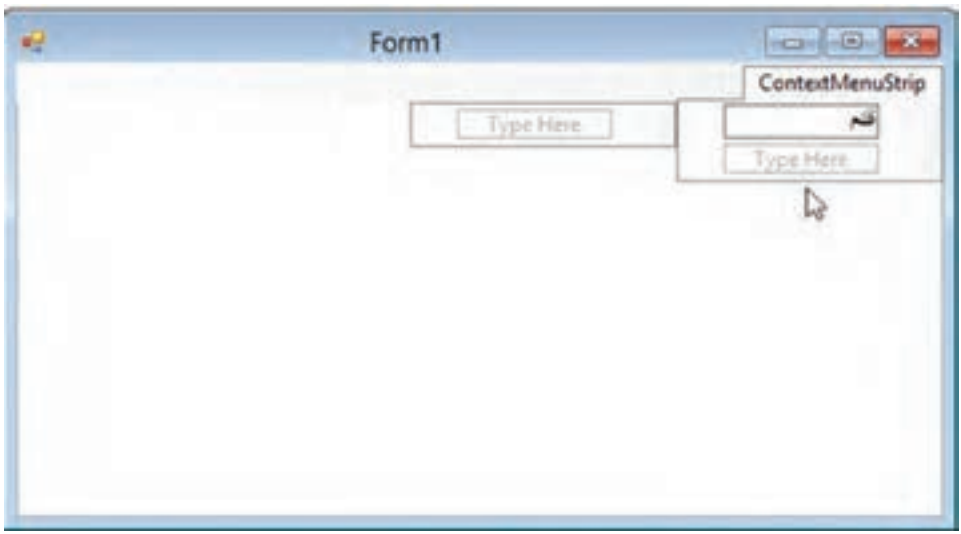
۳ــ درج کنترل: روی کنترل ContextMenuStrip در جعبه ابزار دوبار کلیک می‌کنیم تا در کلاس فرم تعریف شود.

۴ــ تنظیم ویژگی‌های منو

جدول ۴ــ۳ــ ویژگی‌های منو

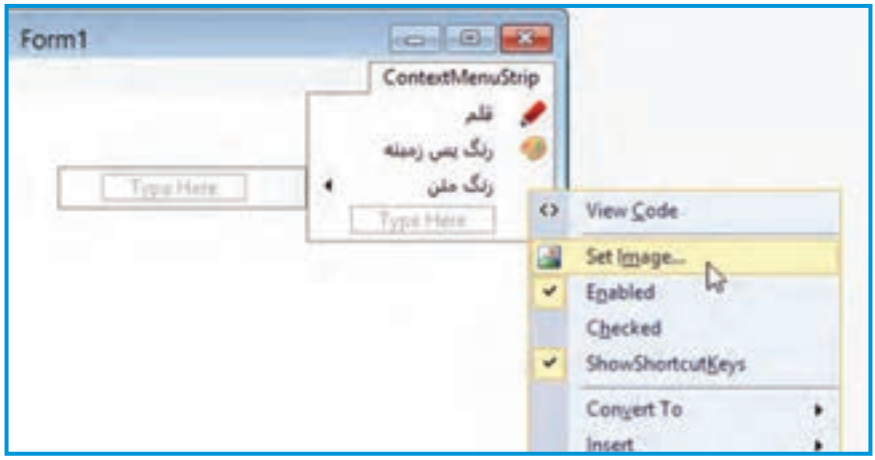
ContextMenuStrip	
ویژگی	مقدار
Name	RightClickMenu
RightToLeft	Yes

۵ــ درج گزینه‌های منو: مشابه آنچه که در بخش MenuStrip گفته شد می‌توانیم گزینه‌های منو را وارد کنیم. نام گزینه قلم را در پنجره properties برابر fontMenuItem قرار می‌دهیم. گزینه‌های «رنگ پس زمینه» و «رنگ متن» را نیز اضافه می‌کنیم و نام آنها را به ترتیب backColorMenuItem و textColorMenuItem قرار می‌دهیم. سپس برای هر گزینه تصویری را انتخاب می‌کنیم.



شکل ۱۷ــ۳ــ درج گزینه‌های منو

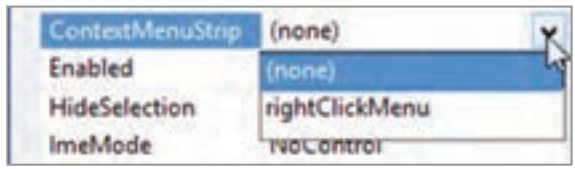
۶ــ نوشتن دستورات برنامه : اکنون نوبت نوشتن دستورهایی است که با کلیک گزینه‌های منو اجرا می‌شوند. برای آنکه متد رویداد کلیک هر گزینه را ایجاد کنیم کافی است که روی آن دوبار کلیک کنیم یا از پنجره properties و قسمت رویدادها، متد رویداد کلیک را ایجاد کنیم.



شکل ۱۸ــ۳ــ کدنویسی برای گزینه‌های منو

```
private void fontMenuItem_Click(object sender, EventArgs e)
{
    fontDialog1.ShowDialog();
    editor.Font = fontDialog1.Font;
}
private void backColorMenuItem_Click(object sender, EventArgs e)
{
    colorDialog1.ShowDialog();
    editor.BackColor = colorDialog1.Color;
}
private void textColorMenuItem_Click(object sender, EventArgs e)
{
     colorDialog1.ShowDialog();
     editor.ForeColor = colorDialog1.Color;
}
```

اکنون برنامه را اجرا کنید. خواهید دید که با کلیک راست روی کادر متنی، منوی طراحی‌شده نمایش داده نمی‌شود. برای اینکه منوی ما به عنوان منوی کلیک راست شناخته شود باید ویژگی ContextMenuStrip برای کادر متنی editor برابر با rightClickMenu شود.



شکل ۱۹ــ۳ــ ویژگی ContextMenuStrip

سؤال: آیا کلیدهای میان‌بر را می‌توان برای گزینه‌های منوی زمینه تعیین کرد؟ قابلیت انتخاب علامت را چطور؟ سؤال : آیا می‌شود به چند کنترل (مانند کادر متنی) یک منوی زمینه را تخصیص داد؟

خودآزمایی فصل سوم

الف) درستی یا نادرستی عبارات زیر را تعیین کنید :

۱ــ گزینه‌های منوی ایجاد شده با Insert standard Items را می‌توان تغییر داد.

۲ــ برای فعال سازی گزینه‌های منو فقط می‌توان از ماوس استفاده کرد.

۳ــ ویژگی Checked برای فعال یا غیر فعال کردن گزینه‌های منو به‌کار می‌رود.

۴ــ از خط جداکننده در طراحی منو برای دسته بندی گزینه‌ها استفاده می‌شود.

۵ــ در ویژوال استودیو امکان ایجاد منوی فارسی وجود ندارد.

ب) جاهای خالی را با عبارت مناسب پر کنید :

۶ــ برای ایجاد نوار منو از کنترل..... استفاده می‌شود.

۷ــ برای فعال سازی یک حرف از نام گزینه به عنوان کلید دسترسی، از علامت ..... قبل از حرف مورد نظر استفاده می‌کنیم.

۸ــ برای تعیین جهت منو از راست به چپ از ویژگی ..... استفاده می‌کنیم.

ج) به سؤالات زیر پاسخ دهید :

۹ــ در چه مواقعی از منوها استفاده می‌شود؟

۱۰ــ کاربرد کلید زمینه را برای گزینه‌های منو بنویسید.

۱۱ــ کاربرد کلید دسترسی برای گزینه منو چیست ؟

۱۲ــ تفاوت دو ویژگی Visible و Enabled را بنویسید.

تمرینات برنامه‌نویسی فصل سوم

۱ــ مثال ۵ ــ ۳ در کارگاه ۲ را طوری تغییر دهید که کادر متنی منبع، دارای یک منو با گزینه «فرستادن به مقصد» و کادر متنی مقصد دارای یک گزینه «گرفتن از منبع» باشد. با زدن هر یک از این گزینه‌ها، عملیات کپی از منبع به مقصد صورت پذیرد.

۲ــ یک پروژه ایجاد کنید و سه تصویر را به وسیلهٔ pictureBox طوری به فرم اضافه کنید که آبشاری روی هم قرار بگیرند. با کلیک راست روی هر تصویر باید منویی ظاهر شود که دارای دو گزینه باشد : «آوردن به جلو» و «فرستادن به عقب»

با انتخاب منوی آوردن به جلو باید تصویر روی سایر تصاویر قرار بگیرد و با انتخاب منوی «فرستادن به عقب» تصویر زیر سایر تصاویر قرار بگیرد.

راهنمایی : برای فرستادن یک کنترل مانند کادر تصویر از متد BringToFront و برای فرستادن به عقب از متد SendToFront استفاده می‌شود. بنابراین برای یک کادر تصویر با نام pictureBox1 داریم :

```
pictureBox1.BringToFront();
pictureBox1.SendToBack();
```

۳ــ برای برنامه بالا یک نوار منو با استفاده از کنترل MenuStrip و مطابق راهنمای صفحه بعد بسازید.

در این منو تصویرهایی که نمایش داده می‌شوند تیک خورده‌اند. با کلیک روی هر گزینه، اگر در حالت فعلی تیک خورده باشد، تصویر مربوط به آن پنهان می‌شود و تیک آن گزینه نیز برداشته می‌شود و اگر در حالت فعلی تیک ندارد برعکس این عمل انجام می‌شود.	
در این منو، تصویری که باید در جلوی همه باشد تیک خورده است. همچنین اگر یک گزینه که تیک ندارد کلیک شود، تصویر مربوط به آن به جلو می‌آید و خود آن گزینه تیک دار می‌شود و گزینه قبلی از حالت تیک دار خارج می‌شود (در کد، همه گزینه‌ها جز گزینه فعلی را از حالت Checked خارج می‌کنیم).	

فعالیت

تکمیل پروژه

برای تکمیل پروژه، به جز جستجو، نیاز به امکانات دیگری نیز داریم. **در این بخش** برای آنکه ظاهر فرم شلوغ نشود وهمچنین کاربرپسند باشد، رابط کاربری طراحی می‌کنیم که متداول است و در انواع واژه نامه‌ها دیده می‌شود. استفاده از منو سبب می‌شود دسترسی به امکانات از این طریق فراهم شود.

مرحله سوم :

۱ــ یک menuStrip به فرم اضافه شود و گزینه‌های آن به صورتی که در شکل زیر مشخص است تعیین شود و ویژگی name و Text گزینه‌ها مطابق جدول زیر تعیین شود.

ویژگی‌ها

ویژگی	مقدار
name	text
fileMenuItem	فایل
registerAndEditMenuItem	ثبت و ویرایش واژه
saveMenuItem	ذخیره جستجو
closeMenuItem	بستن

۲ــ همچنین یک منوی زمینه بـا عـنـوان «ذخیره جستجو» بـــرای کـــادر معنـی و تـوضیح، طراحـی و تعیین شود (شکل روبـه‌رو). نام این منو را saveMenuItem2 قرار می‌دهیم.

۳ــ پس از ایجاد منوهای بالا برای گزینه بستن ، کد متد EH رویداد کلیک آن، نوشته شود.

```
private void closeMenuItem_Click (Object sender, EventArgs e)
{
    this.Close ( );
}
```

۴ــ برای گزینه «ثبت و ویرایش واژه» متد EH رویداد کلیک را این‌گونه تعیین کنید.

```
private void registerAndEditMenuItem_Click (Object sender, EventArgs e)
{
    MessageBox.Show ("(: منتظرم ،تا ایجاد یک راه‌حل مناسب در فصل‌های بعدی");
}
```

اگر دکمه بستن زده شود، فرم بسته خواهد شد و اگر دکمه «ثبت و ویرایش» زده شود پیام آن نمایش داده می‌شود.

برای گزینه «ذخیره جستجو» نیز کد مورد نیاز را بنویسید تا همین پیام نمایش داده شود.

توجه : برای گزینه «ذخیره جستجو» در هر دو منو از یک متد EH استفاده کنید. در این برنامه، این متد را saveMenuItem_Click نام‌گذاری کنید.

واژگان و اصطلاحات انگلیسی فصل سوم

ردیف	واژه انگلیسی	معنی به فارسی
۱	Backcolor	
۲	Checked	
۳	Context	
۴	Copy	
۵	Destination	
۶	Dialog	
۷	Enabled	
۸	Enter	
۹	Forecolor	
۱۰	Hot Key	
۱۱	Image	
۱۲	Insert	
۱۳	Item	
۱۴	Menu	
۱۵	Paste	
۱۶	Separatore	
۱۷	ShortcutKey	
۱۸	Source	
۱۹	Strip	
۲۰	Tool bar	
۲۱	Tools	
۲۲	Visible	