

رویدادهای ماوس و صفحه کلید


در بیشتر برنامه‌های 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 برنامه‌سازی 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.

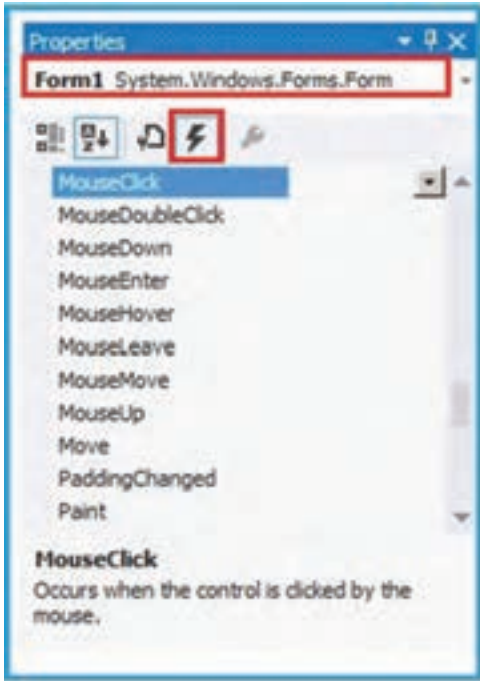
پس از پایان این فصل انتظار می‌رود که فراگیر بتواند :

- ۱- رویدادهای مربوط به ماوس و صفحه کلید را توضیح دهد.
- ۲- از رویدادهای مربوط به ماوس در برنامه‌های کاربردی تعاملی^۱ استفاده نماید.
- ۳- رویدادهای مربوط به صفحه کلید را در برنامه‌های خود به کار بندد.

۲-۱- 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 ۲-۱). عملکرد هر رویداد در جدول ۲-۱ آورده شده است.



شکل ۲-۱- رویدادهای ماوس

جدول ۲-۱

ردیف	نام رویداد	شرح رویداد
۱	MouseDown	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

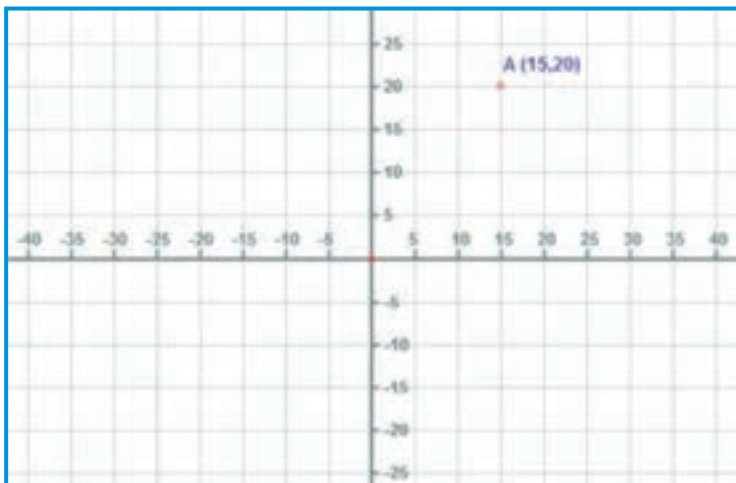
هنگامی که به وسیلهٔ ماوس بر روی فرم کلیک می‌کنید رویداد MouseClick و در صورتی که ماوس را حرکت دهید، رویداد MouseMove رخ می‌دهد. مثال زیر را برای آشنایی با رویداد MouseMove ببینید.

مثال ۲-۱: برنامه‌ای بنویسید که با حرکت ماوس بر روی فرم، موقعیت ماوس با دو عدد طول و عرض نشان داده شود.



شکل ۲-۲- مختصات ماوس

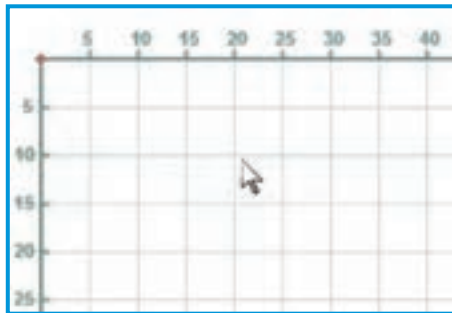
الگوریتم یا روش انجام کار: از درس ریاضیات به خاطر دارید که هر نقطه در صفحه با دو عدد طول و عرض، نسبت به مبدأ نشان داده می‌شود. به آن دو عدد، مختصات نقطه^۱ می‌گویند. شکل ۲-۳ موقعیت نقطه A در صفحه را نشان می‌دهد.



شکل ۲-۳- موقعیت نقطه در صفحه

در VS نقطه مبدأ برای تعیین مختصات یک شیء یا ماوس بر اساس گوشه بالا سمت چپ در نظر گرفته می‌شود (شکل ۲-۴).

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 و عدد 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 است.



شکل ۲-۴- مبدأ مختصات در صفحه

سؤال؟ مقدار X و Y اشاره گر ماوس در شکل ۲-۴ چقدر است؟

هنگامی که ماوس را حرکت می‌دهید، دو عدد X و Y به وسیله 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:

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

۲- تعیین ویژگی‌های فرم: ویژگی‌های فرم را به صورت جدول ۲-۲ تنظیم کنید.

جدول ۲-۲- ویژگی های فرم

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

۳- تعیین ویژگی های برجسب ها : دو کنترل برجسب بر روی فرم قرار دهید و ویژگی آنها را مطابق جداول زیر تعیین کنید.

جدول ۲-۴- ویژگی های کنترل برجسب دوم

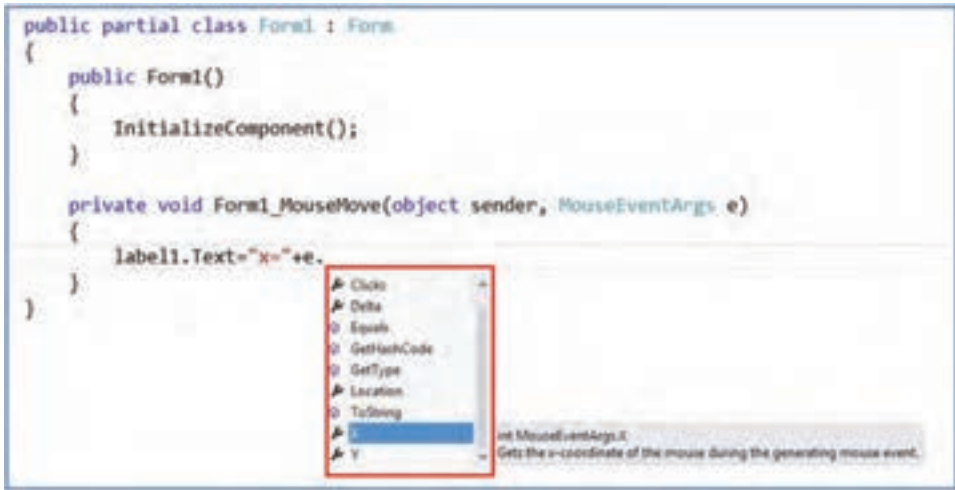
Label		
ویژگی		مقدار
Name		Label2
Text		
AutoSize		True
Location	X	150
	Y	240

جدول ۲-۳- ویژگی های کنترل برجسب اول

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

۴- 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 وسیله کنترل برجسب label1 نشان دهیم.

به وسیله پارامتر e در متد MouseMove، می‌توانیم به جزئیات رویداد ماوس دسترسی داشته باشیم مثلاً می‌توانیم موقعیت ماوس را ببینیم. شکل ۲-۵ منوی IntelliSense ویژگی‌های ماوس را نشان می‌دهد. ویژگی X مقدار طول و ویژگی Y مقدار عرض موقعیت ماوس را در اختیار برنامه‌نویس قرار می‌دهد.



شکل ۲-۵

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.

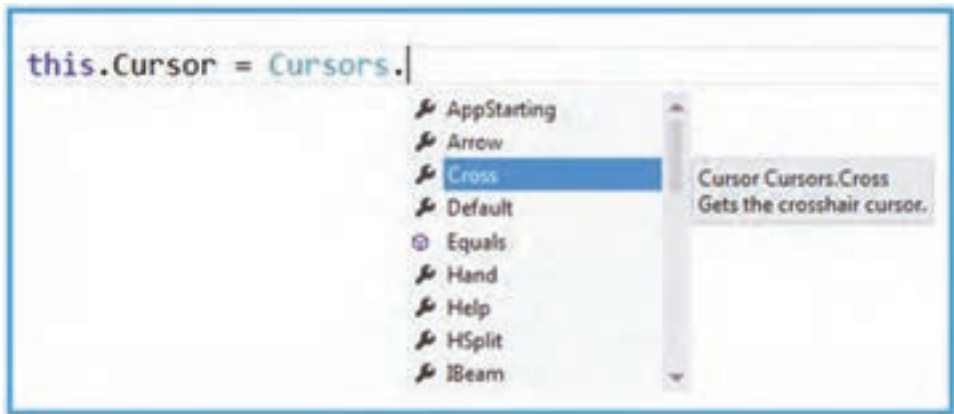
در کتابخانه .NET، کلاسی به نام Cursors تعریف شده که دارای تعدادی ویژگی استاتیک است که هر یک از آنها، شکل مختلفی از علامت ماوس را در اختیار قرار می‌دهد. به‌عنوان نمونه، ویژگی Cross شکل ظاهری یا علامتی از ماوس، به صورت دو خط متقاطع^۱ (به شکل +) فراهم می‌کند.

^۱ - 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 ۶-۲ اضافه می‌کنیم.



شکل ۶-۲- 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 ویزوال 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 برجسب‌ها قرار می‌گیرد شکل اشاره‌گر ماوس به حالت عادی فلش تغییر پیدا کند. برای این کار می‌توانیم از دو روش استفاده کنیم:

۱- نوشتن کد به وسیله VS: اگر ویژگی Cursor هر دو برجسب را در پنجره Properties برابر Default قرار دهیم کدها به وسیله VS نوشته می‌شود.

۲- 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;
```

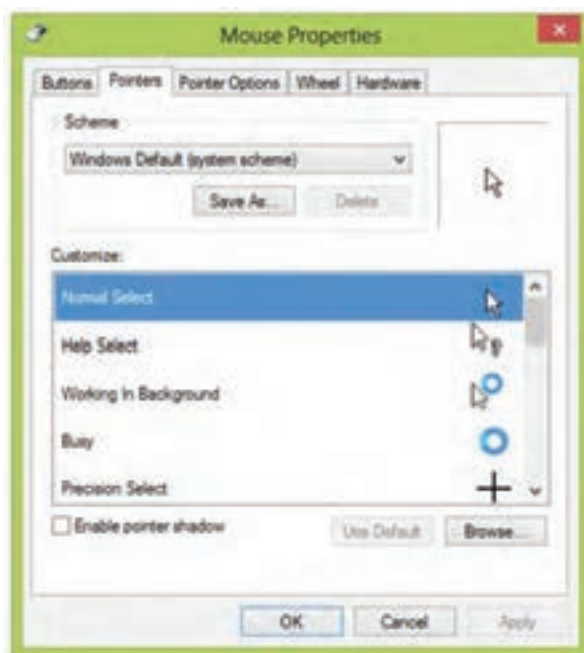
```
label2.Cursor = Cursors.Default;
```

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

نکته

شکل ظاهری ماوس که به وسیله ویژگی‌های کلاس Cursors فراهم می‌شود، مطابق با شکل‌های تعیین شده برای ماوس، در سیستم عامل است. همان‌طور که می‌دانید به وسیله تنظیمات ماوس در کنترل پنل سیستم عامل 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 ۲-۷).



شکل ۲-۷

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 ۲- تونل (راهرو) ماوس

مثال ۲-۲: 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 وگرنه بازی با شکست مواجه می‌شود. چگونه این برنامه را طراحی کنیم؟ در این برنامه خواهید دید به راحتی، سازنده یک بازی خواهید شد.

الگوریتم یا روش انجام کار: قبل از هر کاری نقشه 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:

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

۲- تعیین ویژگی‌های فرم

جدول ۲-۵- ویژگی‌های فرم

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

۳- تعیین ویژگی‌های برجسب منطقه پیروزی

جدول ۲-۶- ویژگی‌های برجسب

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



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 ۲-۸).

شکل ۲-۸- تعیین ویژگی TextAlign

پس از انجام این تنظیم‌ها می‌توانید اندازه برجسب 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.

۴- تعیین ویژگی‌های برجسب راهروها

جدول ۲-۷- ویژگی‌های برجسب

Label	
ویژگی	مقدار
Name	Label1
Text	
AutoSize	false
BackColor	#ff8000

با انجام چند عملیات Copy و Paste ساده، سایر برجسبها را از روی برجسب اولیه نسخه برداری کنید و با جابه جا کردنشان، یک راهرو برای عبور ماوس بسازید (مانند شکل ۹-۲).



شکل ۹-۲- فرم بازی تونل ماوس

۵- ایجاد واکنش نسبت به حرکت ماوس: در این برنامه اگر ماوس از راهرو امن خود بیرون رود بازی با شکست مواجه می شود. 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 می توانیم از آن استفاده نماییم. خوشبختانه این رویداد برای کنترل های مختلف از جمله فرم وجود دارد. این رویداد MouseEnter نام دارد.

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

```
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 ۱۰-۲- 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 برجسب goal (برجسب 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 برجسب goal کلیک کنیم تا انتخاب شود. در پنجره Properties دکمه Events را انتخاب می‌کنیم، سپس در گروه Mouse رویداد MouseEnter را پیدا کرده و نام Victory را در آن می‌نویسیم. 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 توانسته اید 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 به جای رویداد MouseEnter فرم، در رویداد MouseLeave کنترل‌های برجسب راهرو نوشته می‌شود. یا اینکه منطقه مجاز حرکت ماوس را فرم و منطقه شکست آن را راهروها در نظر گرفت.

توسعه و بهبود برنامه

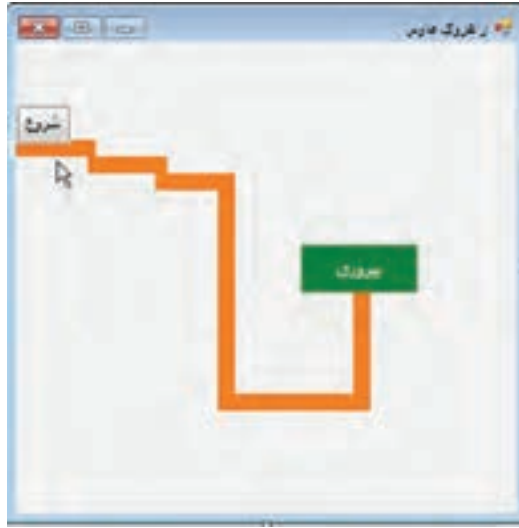
با انجام بازی، خواهیم دید هر ورود ماوس به فرم، ایجاد پیام شکست می‌کند. حتی اگر بازی را پیروز شده باشیم به محض آنکه دوباره ماوس را روی فرم بیاوریم پیام شکست نمایش داده می‌شود. این موضوع برای منطقه پیروزی هم صادق است. برای رفع این مشکل چه باید کرد؟ می‌توانیم از یک دکمه استفاده نماییم تا با فشردن آن بازی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

هنگامی که از دو حالت نام می‌بریم 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 ۱۱-۲- 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؟

۲-۲- 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 وسیله کاربر زده شده است. در روی صفحه کلید، کلیدهایی وجود دارند که با فشردن آنها علامتی یا کاراکتری روی مانیتور ظاهر می‌شود، به این کلیدها یا کاراکترها، کاراکترهای چاپ‌شدنی^۱ می‌گویند. اما بعضی کلیدها نظیر Ctrl، ALT، Shift، Home، End و کلیدهای تابعی مانند F1 تا F12، علامتی را روی صفحه نشان نمی‌دهد. خوشبختانه به وسیله رویدادهای ایجاد شده، می‌توان 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 از صفحه 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 وسیله کاربر فشار داده شده است. اما از فشردن اکثر کلیدهای

^۱ - Printable Characters

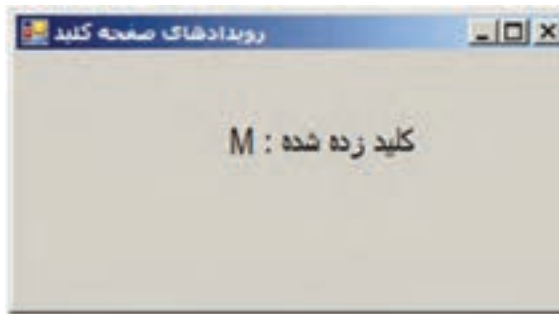
غیرچاپی نمی‌توان مطلع شد و برای این منظور در دو رویداد دیگر KeyUp و KeyDown از فشردن این کلیدها می‌توان خبردار شد.

جدول ۸-۲

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.

مثال ۲-۳: می‌خواهیم برنامه‌ای بنویسیم که با فشردن کلیدی، کاراکتر مربوطه را با استفاده از یک برجسب نمایش دهد (شکل ۱۲-۲).



شکل ۱۲-۲

الگوریتم یا روش انجام کار: با توجه به توضیحاتی که در مورد رویدادهای صفحه کلید داده شد، کافی است یک برجسب 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 برجسب نمایش دهیم. بدین منظور مراحل زیر را دنبال می‌کنیم:

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

۲- تعیین ویژگی‌های فرم: ویژگی‌های فرم را طبق جدول زیر تعیین کنید.

جدول ۹-۲- ویژگی‌های فرم

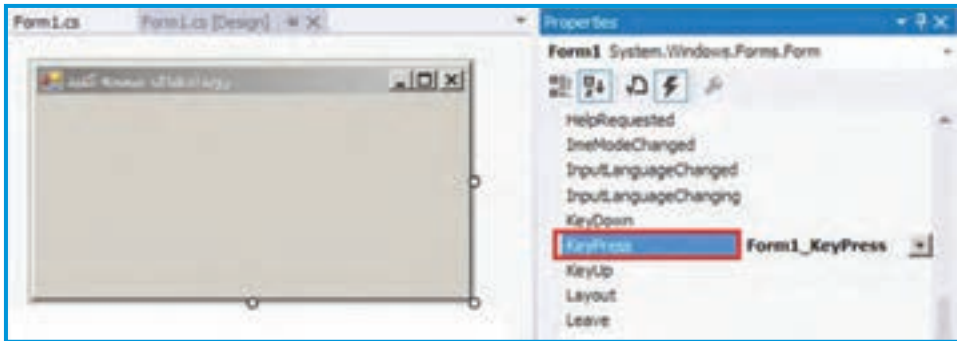
Form		
ویژگی	مقدار	
Name	Form1	
Text	رویدادهای صفحه کلید	
Size	Width	300
	Height	200

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

جدول ۱۰-۲- ویژگی‌های برچسب

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

۴- ایجاد واکنش نسبت به صفحه کلید: مطابق شکل ۱۳-۲ 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 ۱۳-۲- ویژگی های 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، می توان کاراکتر متناظر با کلید زده شده را به دست آورد، که آن را از طریق برجسب نمایش می دهیم. اکنون برنامه را اجرا کنید و کلیدهای مختلفی را بزنید و نمایش آن را روی برجسب مشاهده کنید. کلیدهای غیرچاپی را نیز آزمایش کنید. آیا کاراکتری نشان داده می شود؟ آیا برنامه فعلی قادر به تشخیص آن می باشد.

سؤال: اگر در کد بالا برای نمایش کاراکتر وارد شده در برجسب، قبل از نمایش ویژگی e.KeyChar، پیام قرار داده نشود، خطا پدید می آید (شکل ۱۳-۲). چگونه می توان این خطا را رفع کرد؟

```
private void Form1_KeyPress(object sender, KeyPressEventArgs e)
{
    label1.Text = e.KeyChar;
}
```

شکل ۱۴-۲. 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 چایی از رویداد `KeyDown` استفاده می کنیم :

مثال ۴-۲ : برنامه مثال قبل را توسعه دهید به طوری که قادر به تشخیص کلیدهای غیر چایی

نیز باشد.

مراحل زیر را برای انجام این کار دنبال می کنیم :

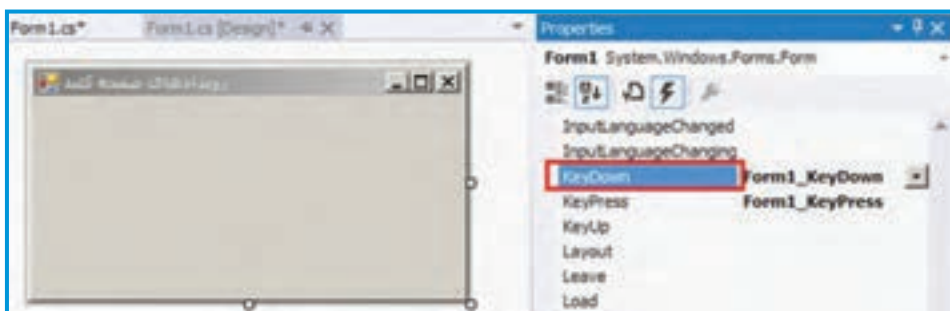
۱- پروژه مثال قبل که برای تشخیص کلید فشار داده شده بود را باز کنید.

۲- یک برچسب دیگر به فرم اضافه کنید. این برچسب برای اطلاع از اینکه کلید چایی یا غیر چایی فشار داده شده به کار می رود.

جدول ۱۱-۲- ویژگی های برچسب

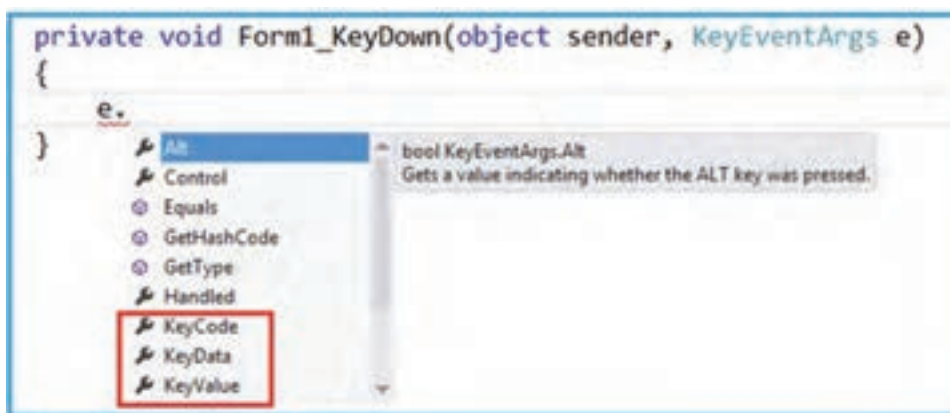
Label		مقدار
ویژگی		Label2
Name		
Text		
AutoSize		True
RightToLeft		Yes
Font		Times New Roman Size: 14
Location	X	50
	Y	80

۳- مطابق شکل ۲-۱۴ 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 ۲-۱۴

۴- متد EH رویداد KeyDown دارای پارامتری به نام e از نوع KeyEventArgs است که دارای ویژگی‌های مختلفی است که در شکل ۲-۱۵ به وسیله منوی IntelliSense تعدادی از ویژگی‌های مختلف آن لیست شده است. در جدول ۲-۱۲ کاربرد هر یک از ویژگی‌ها را مشاهده می‌کنید:



شکل ۲-۱۵- ویژگی‌های پارامتر e

جدول ۱۲-۲- 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 - 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 در متد EH رویداد KeyDown مطابق با شکل می‌نویسیم تا با مقدار خروجی و طریقه استفاده از ویژگی‌های مختلف جدول ۱۲-۲ آشنا شویم.

```
private void Form1_KeyDown (object sender, KeyEventArgs e)
```

```
{
    label2.Text = "KeyData: " + e.KeyData + "\n" +
                 "KeyCode: " + e.KeyCode + "\n" +
                 "KeyValue: " + e.KeyValue + "\n";
}
```



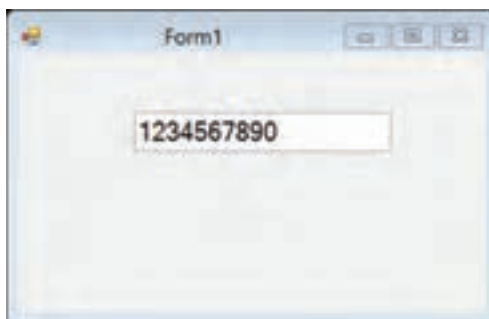
اکنون برنامه را اجرا کرده و کلیدهای مختلفی از هر دو گروه چایی و غیرچایی را بزنید و عکس‌العمل برنامه را مشاهده کنید. مثلاً اگر کلید shift را نگه داشته و حرف P را بزنید خروجی برنامه چنین خواهد شد:

شکل ۱۶-۲

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

کار در کارگاه ۳: سافت یک کادر متنی عددی

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



شکل ۱۷-۲

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

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

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

جدول ۱۳-۲- ویژگی‌های کادر متن

TextBox		
ویژگی		مقدار
Name		input
Font	Size	14

۳- 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:

- ۱- متد `IsDigit` برای ساختار `char` مشخص می‌کند کراکتر ورودی عدد است یا خیر؟ به عنوان نمونه جواب `char.IsDigit('1')` برابر `true` خواهد بود.
- ۲- توسط دستور `(char)e.KeyCode` تبدیل نوع کد کلید به حرف متناظر صورت می‌پذیرد.
- ۳- 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؟
- ۴- با عملگر نفیض "!" که پشت کل عبارت منطقی شرط آمده است مشخص می‌کنیم اگر عبارت منطقی ما درست نباشد (کراکتر 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 نمی کنند چرا؟ با استفاده از نوع شمارشی `Keys` و مقادیر `Keys.Right` و `Keys.Left` می توانیم 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؟

الف) درستی یا نادرستی عبارات زیر را تعیین کنید :

- ۱- برای ایجاد واکنش نسبت به حرکت ماوس از رویداد MouseMove استفاده می شود.
- ۲- با استفاده از ویژگی Cursor در کنترل ها می توان شکل ماوس را تغییر داد.
- ۳- با استفاده از ویژگی e.KeyChar در رویداد KeyPress می توان فهمید کدام کلید فشار داده شده است.
- ۴- با استفاده از رویدادهای KeyDown و KeyUp نمی توان از فشردن کلیدهای غیرچاپی مطلع شد.
- ۵- اگر مقدار ویژگی SppressKeyPress یک کنترل False باشد، مانع از اجرا و تأثیر کلید روی کنترل می شود.
- ب) جاهای خالی را با عبارت مناسب پر کنید :
- ۶- به وسیله پارامتر e.X و e.Y در متد MouseMove، می توانیم به ماوس دسترسی داشته باشیم.
- ۷- برای دسترسی به کد کلید فشار داده شده در رویداد keydown از ویژگی پارامتر e استفاده می شود.
- ۸- اگر بخواهیم ببینیم چه حرفی فشار داده شده، باید از رویداد استفاده کنیم.
- ج) جدول زیر را تکمیل کنید.

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

د) به سوالات زیر پاسخ دهید.

۱۳- دستوری بنویسید که در رویداد KeyPress کنترل کادر متنی، کد عددی کاراکتر وارد

شده در کنترل برجسب نمایش داده شود.

۱۴- برای نمایش کد کلید shift از کدام رویداد صفحه کلید می توان استفاده کرد؟

تمرینات برنامه‌نویسی فصل دوم

۱- 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.



۲- 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:

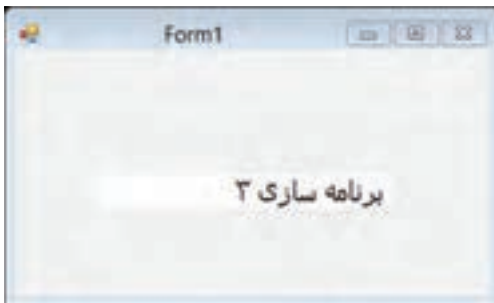
بالا : افزایش قد (+۱۰)

پایین : کاهش قد (-۱۰)

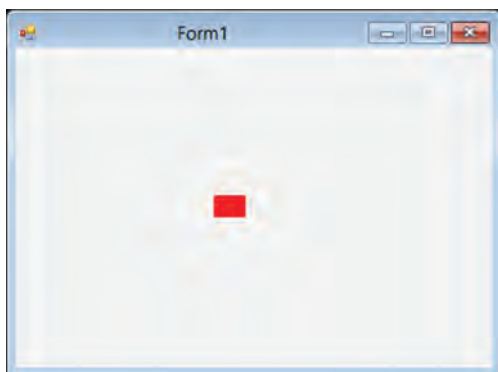
راست : افزایش عرض (+۱۰)

چپ : کاهش عرض (-۱۰)

۳- 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).



۴- 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.



۵- یک بازی دو نفره طراحی کنید که نفر اول، با دابل کلیک اشاره گر ماوس را پنهان کند. آنگاه بازیکن دوم تا ۵ بار فرصت داشته باشد روی محل هدف کلیک کند. در صورت 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 اشاره گر ماوس از

دستور `Cursor.Hide()` و برای ظاهر شدن آن از دستور `Cursor.Show()` استفاده کنید.

۶- پروژه ای بسازید که به عنوان 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 می شود. اگر به زمین برخورد کند یک امتیاز کم می شود. 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 سفینه ها پنج عدد است.

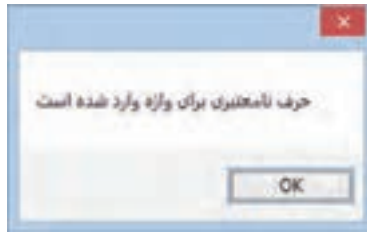


تکمیل پروژه

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:

- ۱- کاربرد 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.



- ۲- 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.



خروج ماوس



ورود اشاره‌گر ماوس

۳- 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 پس‌زمینه 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 برنامه‌نویسی 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 پس زمینه 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();
}
```


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	

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 تفکیک‌ناپذیر هر برنامه خوب محسوب می‌شوند و راهی راحت و پرکاربرد برای دسترسی کاربر به تمام قسمت‌های برنامه را فراهم می‌کنند. برای مثال برنامه ویرال استودیو با استفاده از منوها، به برنامه‌نویس 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:

- ۱- دلایل استفاده از منو را بیان کند و منوی مناسبی برای برنامه خود تعریف نماید.
- ۲- منوهای نواری و زمینه‌را در واسط گرافیکی کاربر پسند مورد استفاده قرار دهد.
- ۳- تنظیمات ویژه‌ای روی منو و گزینه‌های آن انجام دهد.

۱-۳- استفاده از منو^۱ 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:

۱- وارد کردن یک کلید خاص مانند کلیدهای دسترسی^۲ و یا میانبر^۳

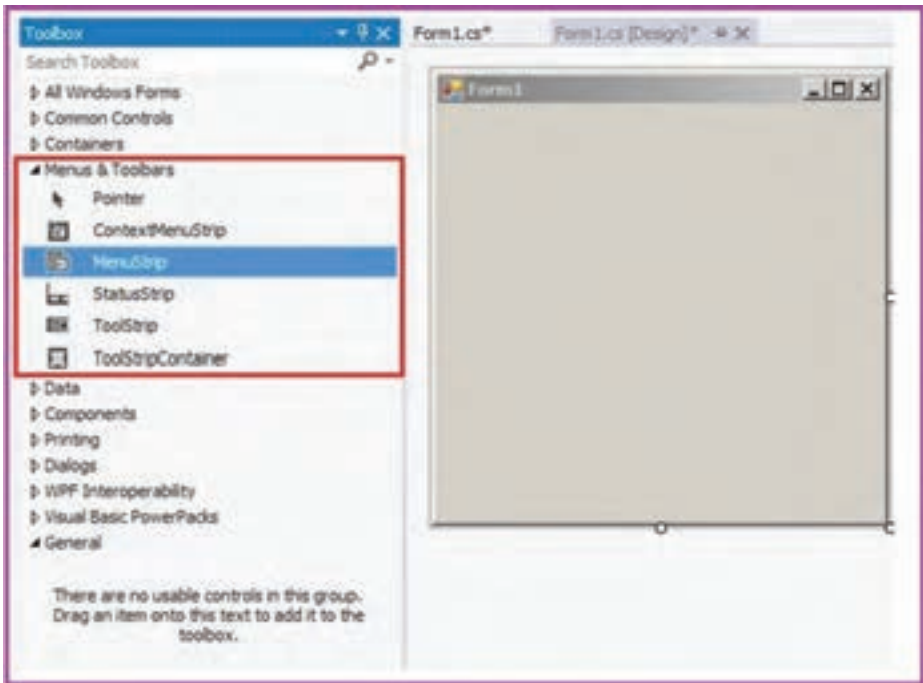
۲- انتقال مکان نما به گزینه مورد نظر و فشردن کلید Enter یا کلیک ماوس

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

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

۲- 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 ۱-۳).



شکل ۱-۳- نحوه دسترسی به ابزار منو

۱- Menu

۲- Hot Key

۳- Shortcut Key

- از نظر ظاهری با اضافه کردن ابزار منو (MenuStrip) به فرم، دو اتفاق می افتد :
- ۱- یک شیء به نام menuStrip1 در زیر فرم قرار می گیرد.
 - ۲- در زیر خط عنوان پنجره فرم، نوار منویی قرار می گیرد که در داخل آن می توانید گزینه های منوهای افقی، عمودی و زیر منوهای مورد نظر خود را تایپ کنید (شکل ۲-۳).



شکل ۲-۳- نحوه کار با شیء منو

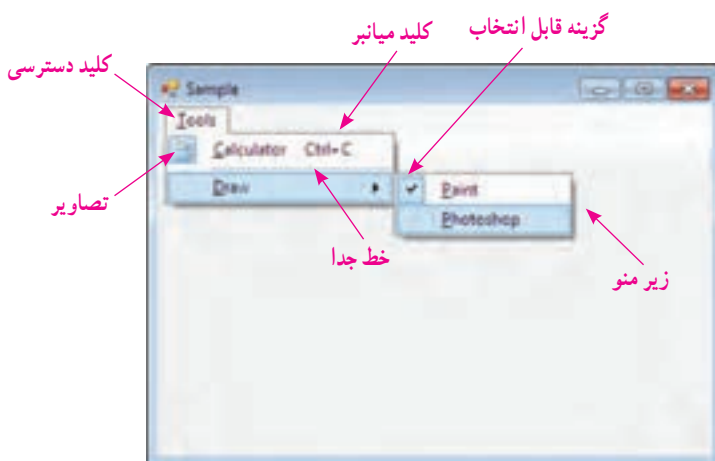
- ۳- در اولین عنوان، عبارت Type Here ظاهر می شود، برای تغییر نام، روی این عنوان کلیک کرده، متن جدید را تایپ و کلید Enter را فشار دهید. با استفاده از کلیدهای مکان نما یا ماوس می توان بین گزینه های منو حرکت کرده، گزینه مورد نظر را انتخاب کرد و عنوان آن را تغییر داد.
- ۴- زمانی که بر روی شیء menuStrip1 کلیک می کنیم، پنجره properties، ویژگی های آن منو را نمایش می دهد و مانند هر کنترل دیگری می توانید ویژگی های آن را تنظیم کنید (جدول ۱-۳).

جدول ۱-۳ ویژگی های متداول شیء منو

ویژگی	شرح
Checked	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 گزینه های منو در کد نویسی
RightToLeft	راست به چپ کردن گزینه های منو (مانند زبان فارسی)
ShortCutKeys	تعریف کلید میانبر
Text	گزینه های منو
Visible	نمایش گزینه های منو

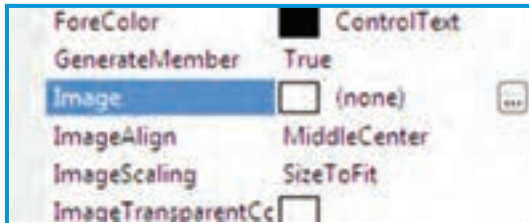
۲-۳- امکانات منو

برای منوهایی که به وسیله این کنترل ایجاد می کنید علاوه بر عنوانی که می توانید برای هر منو و گزینه های منو نمایش دهید، می توانید امکاناتی شامل آیکن (تصویر)، کلید دسترسی، کلید میانبر و یا حتی گزینه هایی با امکان قابلیت انتخاب را به کار گیرید (شکل ۳-۳).



شکل ۳-۳- امکانات منو

۱-۲-۳ جدا کننده بین گزینه‌ها : جداکننده‌ا به خط افقی گفته می‌شود که با قرار گرفتن در بین گزینه‌های منو، آنها را دسته‌بندی می‌کند. برای اضافه کردن جداکننده به گزینه‌های منو، باید در هنگام طراحی منو، در قسمت نام گزینه منو علامت - (منها) قرار داد.

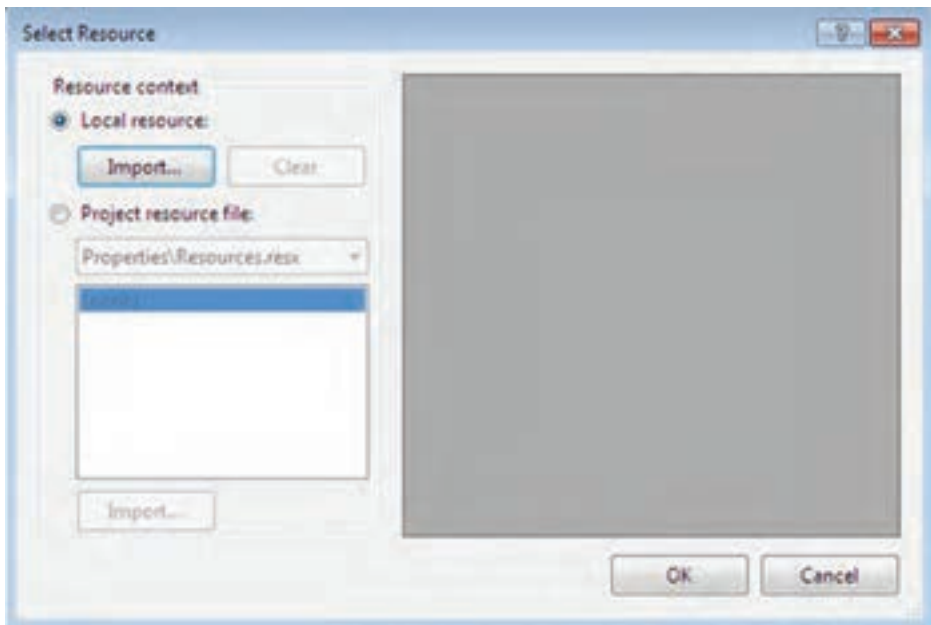


۲-۲-۳ اضافه کردن آیکن

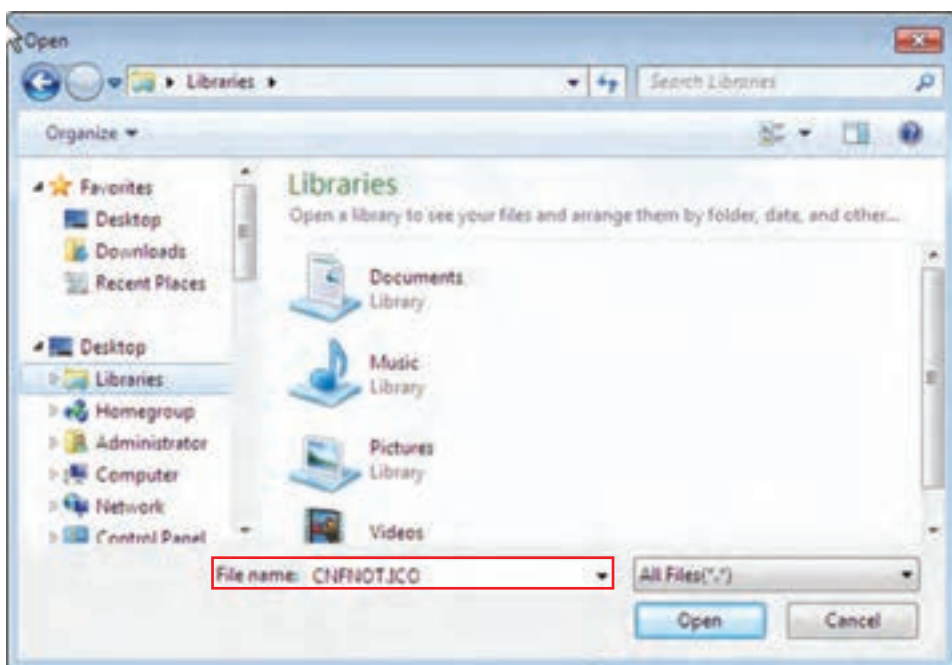
به منوها : شما می‌توانید برای هر گزینه منو، یک تصویر کوچک را انتخاب کنید. که در سمت چپ آن نمایش داده شود برای این کار از خاصیت Image استفاده می‌شود. گزینه مورد نظر در منو

شکل ۳-۴- تعیین ویژگی Image گزینه منو

را انتخاب نمایید. بر روی دکمه کوچک کنار خاصیت Image در پنجره Properties کلیک کنید. منبعی که عکس در آن قرار دارد را انتخاب و سپس مسیر آن را مشخص نمایید. اگر اندازه عکس بزرگ باشد، به صورت خودکار بر اساس مقدار خاصیت ImageScaling اندازه آن تغییر می‌کند.



شکل ۳-۵ اضافه کردن تصویر به منابع پروژه با استفاده از دکمه Import



شکل ۶-۳- 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 .jpg, .png, .bmp, .gif, .wmf) می‌توان استفاده کرد.

۳-۲-۳- ایجاد کلید دسترسی: کلیدهای دسترسی، آن دسته از کلیدها هستند که برای فعال کردن منو و گزینه‌های آن با استفاده از صفحه کلید تعریف می‌شوند. که با فشردن هم‌زمان Alt و یک حرف خاص منجر به باز شدن منو می‌شوند.
برای ایجاد کلید دسترسی، قبل از کاراکتر مورد نظر از علامت & استفاده می‌کنیم.

&Edit نمایش → Edit

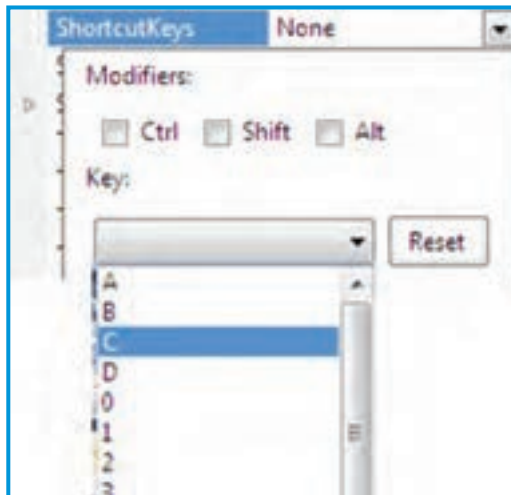
در مثال بالا منوی Edit با استفاده از کلیدهای Alt+E قابل دسترسی است.

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

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

۴-۲-۳- ایجاد کلید میان بر : با استفاده از ویژگی ShortCutKeys، می توان کلید میان بر به منو اضافه کرد که باعث می شوند تا کاربران بتوانند عملکرد گزینه های منو را با سرعت بیشتری انجام دهند.

در پنجره properties در قسمت ShortcutKeys می توان کلید ترکیبی مورد نظر را انتخاب نمود. کلیدهای ترکیبی از ترکیب کلیدهای Ctrl یا Shift یا Alt به همراه حروف الفبا یا ارقام درست می شوند (شکل ۷-۳).



شکل ۷-۳- نحوه ایجاد کلید میان بر

سؤال: در استفاده از کلیدهای میان بر آیا بین حروف کوچک و بزرگ تفاوت وجود

دارد؟

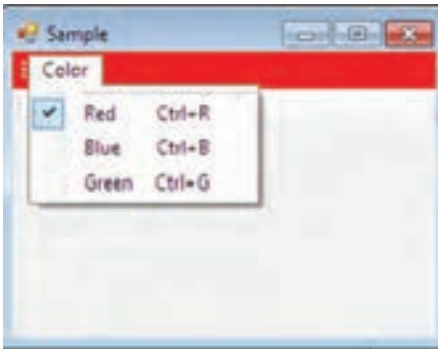
در انتخاب کلید ترکیبی برای کلید میان‌بر دقت کنید تا با کلیدهای میان‌بر ویندوز تداخل نداشته باشد.

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

کار در کارگاه ۱

مثال ۱-۳: فرمی طراحی کنید که با استفاده از گزینه‌های منو، رنگ نوار منو تغییر کند.

- ۱- درج کنترل منو: یک شیء از کنترل MenuStrip برای برنامه ایجاد می‌کنیم.
- ۲- 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 ۸-۳).



شکل ۸-۳- اجرای منوی طراحی شده برای مثال ۱-۱

- ۳- 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 ۳-۲- ویژگی های منو

ویژگی	مقدار		
name	RedToolStripMenuItem	blueToolStripMenuItem	greenToolStripMenuItem
Checked	True	false	false
ShortcutKey	Ctrl+R	Ctrl+B	Ctrl+G
Text	Red	Blue	Green

۳-۳- طراحی منوی استاندارد

علاوه بر اینکه می توانید منوی دلخواه خود را بسازید، اگر مایل باشید می توانید منوی های استاندارد که معمولاً در برنامه های کاربردی ویندوز دیده می شود، به برنامه خود اضافه کنید.

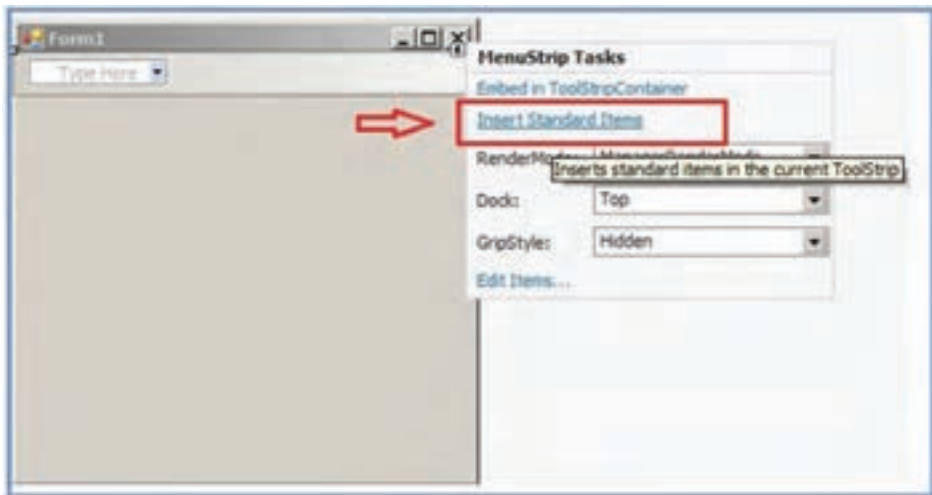
مثال ۳-۲: چگونه منوهای استاندارد به فرم خود اضافه کنیم؟

الگوریتم یا روش انجام کار:

درج کنترل منو: ابتدا کنترل ToolStrip 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 می شود. گزینه Insert Standard Items را انتخاب می کنیم (شکل ۳-۹).



شکل ۳-۹- 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 ۳-۱۰).



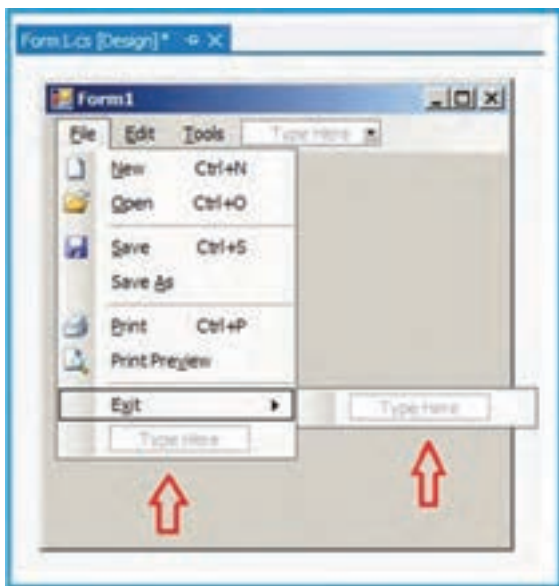
شکل ۳-۱۰ نحوه انتخاب منوی استاندارد با استفاده از کلیک راست روی نوار منو

در هر یک از دو روش، منوی استاندارد به فرم مانند شکل ۳-۱۱ اضافه می‌شود.



شکل ۳-۱۱ 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 ۳-۱۲).



شکل ۳-۱۲- گزینه منوی File از منوی استاندارد

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

۳-۴- 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 برقرار کردن 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 ۳-۳: 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 کدنویسی 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 ۳-۴: می‌خواهیم برای مثال ۱-۲ که طراحی کردیم کدنویسی کنیم به طوری که با انتخاب رنگ مورد نظر، رنگ نوار منو تغییر کند و گزینه مورد نظر علامت‌دار شود. الگوریتم یا روش انجام کار: ۱- 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)
{
    menuStrip1.BackColor = Color.Red; ← تعیین رنگ نوار منو
    colorStripMenuItem.BackColor = Color.Red; ← تعیین رنگ گزینه منو
}
```

```

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

```

۲- 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 ۵-۳: 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:

۱- 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.

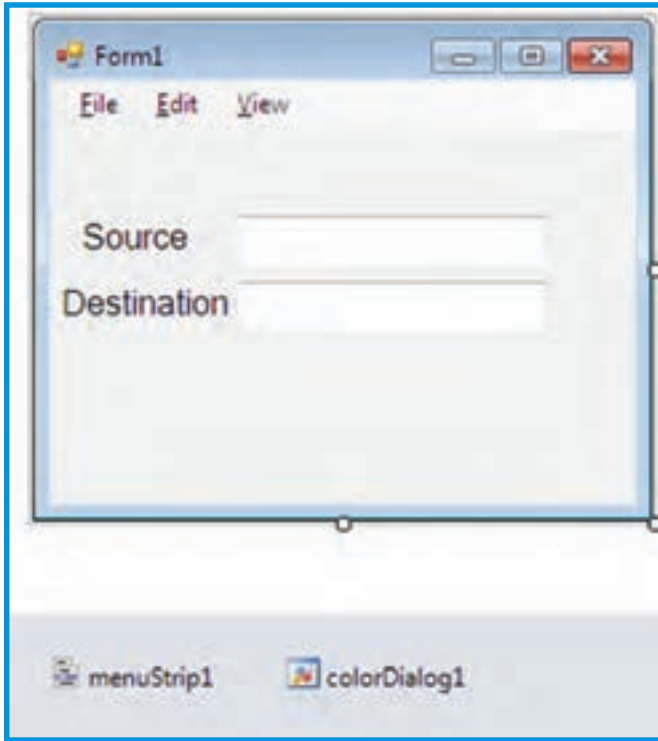
۲- ایجاد گزینه‌های مورد نظر برای منو: در منوی File، گزینه‌های New و Exit را

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.

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

۴- درج کنترل ColorDialog: از قسمت ابزار Dialogs کادر محاوره‌ای ColorDialog

را برای فرم در نظر می‌گیریم (شکل ۱۳-۳ و شکل ۱۴-۳).



شکل ۳-۱۴- فرم مثال ۳-۵



شکل ۳-۱۳- جعبه ابزار Dialogs

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

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

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

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

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

```
string s1;
```


۸- نوشتن رویداد *Click* برای گزینه *Copy* : روی گزینه *Copy* دو بار کلیک کرده و

کد زیر را می‌نویسیم :

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

۹- 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.

۱۰- 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;
```

۱۱- نوشتن رویداد *Click* برای گزینه *ForeColor* : روی گزینه *ForeColor* دو بار

کلیک کرده و کدهای زیر را می‌نویسیم :

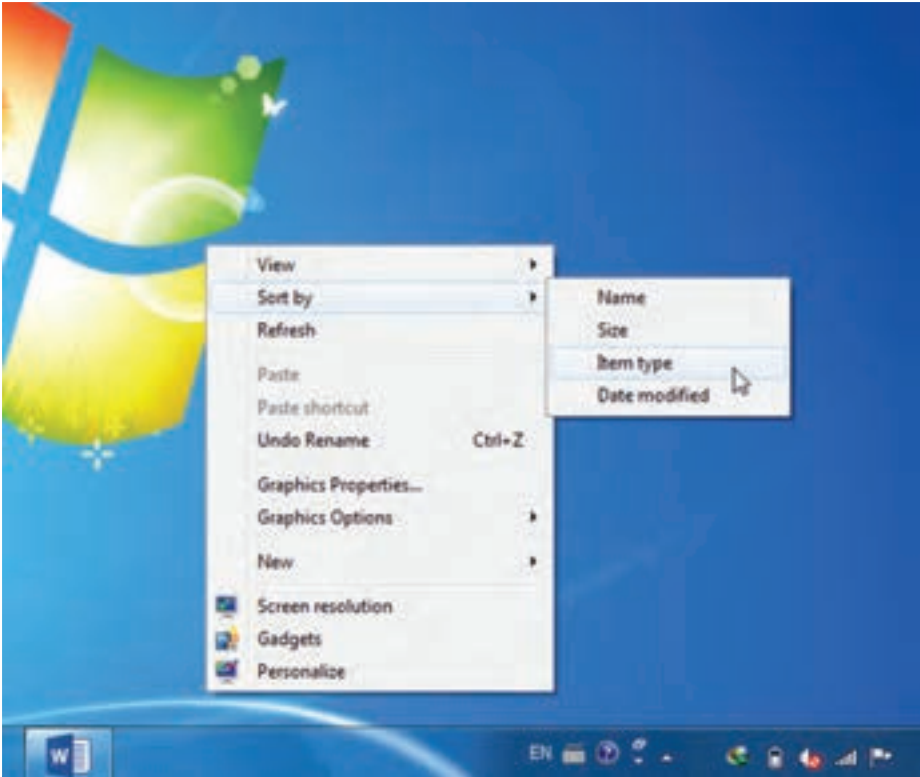
```
colorDialog1.ShowDialog();  
textBox1.BackColor = colorDialog1.Color;  
textBox2.ForeColor = colorDialog1.Color;
```

سؤال: چرا بهتر است برای تغییر گزینه منو به جای تغییر نام، آن گزینه را حذف کرده

و گزینه جدید تعریف شود؟

۳-۵- منوی زمینه^۱

شکل ۳-۱۵ را بارها روی میز کار ویندوز دیده‌اید. این گونه 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 ۳-۱۵- 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 صفحه بعد ببینید.

مثال ۳-۶: 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 ۳-۱۶).



شکل ۳-۱۶- ویرایشگر متنی با منوی زمینه

الگوریتم یا روش انجام کار :

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

۲- تنظیم ویژگی‌های کادر متنی

جدول ۳-۳- ویژگی‌های کادر متنی

Text Box	
ویژگی	مقدار
Name	editor
Dock	fill
Multiline	True

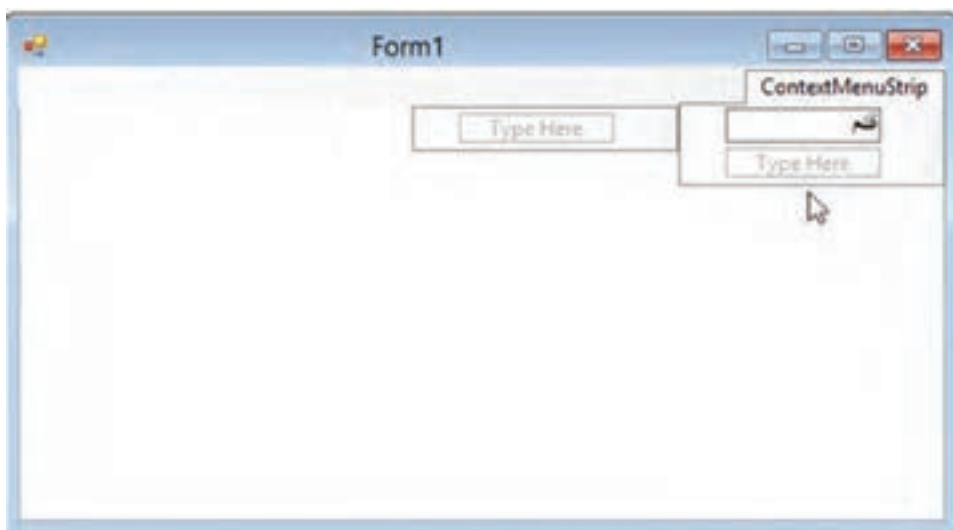
۳- درج کنترل: روی کنترل ContextMenuStrip در جعبه ابزار دوبار کلیک می‌کنیم تا در کلاس فرم تعریف شود.

۴- تنظیم ویژگی‌های منو

جدول ۳-۴- ویژگی‌های منو

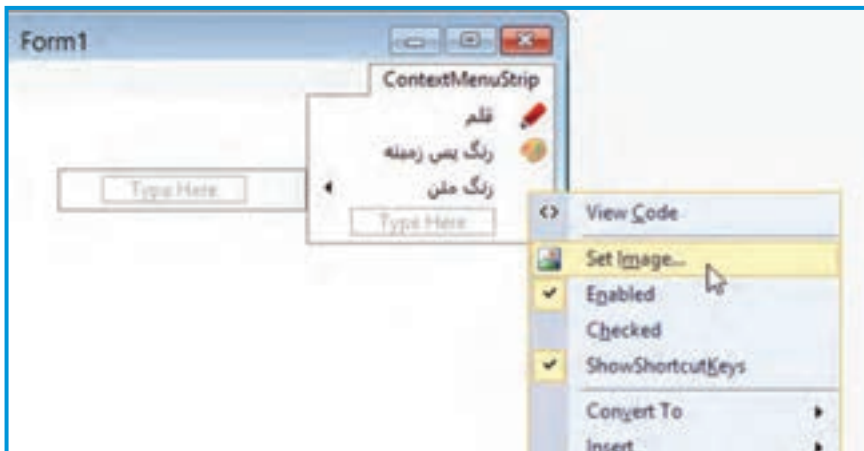
ContextMenuStrip	
ویژگی	مقدار
Name	RightClickMenu
RightToLeft	Yes

۵- 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 ۳-۱۷- درج گزینه‌های منو

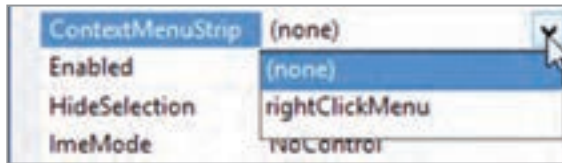
۶- نوشتن دستورات برنامه: اکنون نوبت نوشتن دستورهای است که با کلیک گزینه‌های منو اجرا می‌شوند. برای آنکه متد رویداد کلیک هر گزینه را ایجاد کنیم کافی است که روی آن دوبار کلیک کنیم یا از پنجره properties و قسمت رویدادها، متد رویداد کلیک را ایجاد کنیم.



شکل ۱۸-۳- 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 طراحی شده 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 ۱۹-۳- ویژگی ContextMenuStrip

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

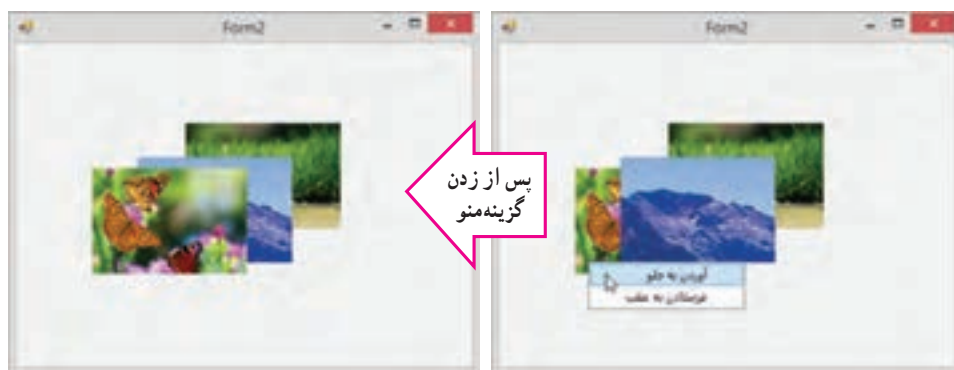
الف) درستی یا نادرستی عبارات زیر را تعیین کنید :

- ۱- گزینه‌های منوی ایجاد شده با Insert standard Items را می‌توان تغییر داد.
 - ۲- برای فعال سازی گزینه‌های منو فقط می‌توان از ماوس استفاده کرد.
 - ۳- ویژگی Checked برای فعال یا غیر فعال کردن گزینه‌های منو به کار می‌رود.
 - ۴- از خط جداکننده در طراحی منو برای دسته بندی گزینه‌ها استفاده می‌شود.
 - ۵- در ویژوال استودیو امکان ایجاد منوی فارسی وجود ندارد.
- ب) جاهای خالی را با عبارت مناسب پر کنید :
- ۶- برای ایجاد نوار منو از کنترل..... استفاده می‌شود.
 - ۷- برای فعال سازی یک حرف از نام گزینه به عنوان کلید دسترسی، از علامت قبل از حرف مورد نظر استفاده می‌کنیم.
 - ۸- برای تعیین جهت منو از راست به چپ از ویژگی استفاده می‌کنیم.
- ج) به سؤالات زیر پاسخ دهید :
- ۹- در چه مواقعی از منوها استفاده می‌شود؟
 - ۱۰- کاربرد کلید زمینه را برای گزینه‌های منو بنویسید.
 - ۱۱- کاربرد کلید دسترسی برای گزینه منو چیست ؟
 - ۱۲- تفاوت دو ویژگی Visible و Enabled را بنویسید.

تمرینات برنامه‌نویسی فصل سوم

۱- مثال ۵-۳ 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 کی‌ی از منبع به مقصد صورت پذیرد.

۲- یک پروژه ایجاد کنید و سه تصویر را به وسیلهٔ pictureBox طوری به فرم اضافه کنید که آشناری 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»

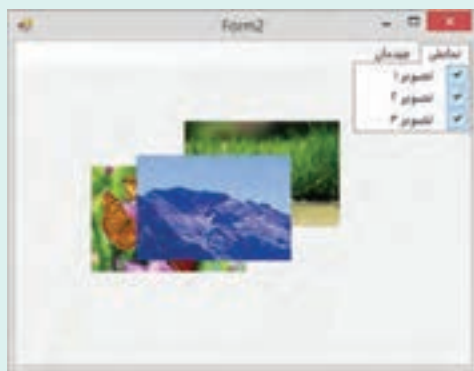


با انتخاب منوی آوردن به جلو باید تصویر روی سایر تصاویر قرار بگیرد و با انتخاب منوی «فرستادن به عقب» تصویر زیر سایر تصاویر قرار بگیرد.

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();
```

۳- 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).

تکمیل پروژه

برای تکمیل پروژه، به جز جستجو، نیاز به امکانات دیگری نیز داریم. در این بخش برای آنکه ظاهر فرم شلوغ نشود و همچنین کاربر پسند 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:

۱- 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

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





۲- همچنین یک منوی زمینه با عنوان «ذخیره جستجو» برای کادر معنی و توضیح، طراحی و تعیین شود (شکل روبه‌رو). نام این منو را saveMenuItem2 قرار می‌دهیم.

۳- 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 ();
}
```

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

```
private void registerAndEditMenuItem_Click (Object sender, EventArgs e)
{
    MessageBox.Show ("تا ایجاد یک راه حل مناسب در فصل های بعدی، منتظرم.");
}
```



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	