فصل اول : مقدمات زبان C#.Net

۱- برخی از ویژگی های زبان C#
۲- انواع داده
۳- متغیرها و طرز تعریف صحیح متغیرها و مقدار دادن به متغیرها
۴- تعریف ثوابت
۵- عملگرها (محاسباتی – رابطه ای – منطقی – ترکیبی – بیتی) + عملگرهای? و کاما ، sizeof
۶- تابع تبدیل یک نوع داده به نوع دیگر در C# (Convert)
۷- نوشتن برنامه با استفاده از کنسول در C# (Console)
۸- برخی توابع مهم مربوط به کنسول
۹- آرگومانهای جایگزین در رشته با استفاده از {}
۱۰- مثال در کلاس
۱۱- تمرین در خانه

۱– برخی از ویژگی های زبان C#

- زبان C# یک زبان شیء گرا ست ، یعنی در این زبان میتوان قطعاتی را ایجاد کرد و در برنامه های مختلف استفاده کرد.
- هر دستور زبان C# به سیمی کلن (;) ختم میشود.
- زبان C# نسبت به بزرگی و کوچکی حروف حساس است.
- توضیحات یا Comments که میخواهیم در برنامه قرار دهیم بایستی بین /* و */ یا بعد از // قرار گیرند.

۲– انواع داده (Data Types)

یکی از مهمترین بخش های هر زبان برنامه نویسی که بایستی به صورت دقیق مورد بررسی قرار گیرد انواع داده ها در آن زبان است. در تمامی زبان های برنامه نویسی داده ها به دو قسمت تقسیم میشوند :

- **الف** - داده های عددی
- **ب** - داده های غیر عددی

الف- داده های عددی به دو دسته داده عددی صحیح و داده عددی اعشاری تقسیم میشوند.

داده صحیح مثل 100 یا 98- و داده اعشاری مثل ۴۰۰.۶۵ و ۰.۰۰۵

جدول زیر انواع داده های عددی صحیح را همراه با تعداد حافظه ای که مصرف میکنند را نشان میدهد:

نوع داده	تعداد بایت	توضیحات
byte	۱	داده صحیح ۸ بیتی
Byte	۱	داده صحیح ۸ بیتی
short	۲	داده صحیح ۱۶بیتی
Int16	۲	داده صحیح ۱۶ بیتی
int	۴	داده صحیح ۳۲ بیتی
Int32	۴	داده صحیح ۳۲ بیتی
long	۸	داده صحیح ۶۴ بیتی
Int64	۸	داده صحیح ۶۴ بیتی
uint	۴	داده صحیح و مثبت (بدون علامت) ۳۲ بیتی
UInt32	۴	داده صحیح و مثبت (بدون علامت) ۳۲ بیتی
ulong	۸	داده صحیح و مثبت (بدون علامت) ۶۴ بیتی
UInt64	۸	داده صحیح و مثبت (بدون علامت) ۶۴ بیتی

جدول زیر انواع داده های عددی اعشاری را همراه با تعداد حافظه ای که مصرف میکنند را نشان میدهد:

نوع داده	تعداد بایت	توضیحات
float	۴	داده اعشاری ۳۲ بیتی
Single	۴	داده اعشاری ۳۲ بیتی
double	۸	داده اعشاری ۶۴ بیتی
Double	۸	داده اعشاری ۶۴ بیتی
decimal	۱۶	داده اعشاری ۱۲۸ بیتی
Decimal	۱۶	داده اعشاری ۱۲۸ بیتی

توجه کنید که در برنامه چه برای داده صحیح و چه برای داده اعشاری به اندازه مورد نیاز برنامه نوع داده را تعریف کنید. مثلاً برای ذخیره کردن سن یک فرد میتوانید از نوع داده byte که یک نوع داده ۱ بایتی است استفاده کرد.

ب– انواع داده های غیر عددی عبارتند از :

داده منطقی – داده کارکتری – داده رشته ای – داده تاریخ و زمان.

جدول زیر انواع داده های غیر عددی را همراه با تعداد حافظه ای که مصرف میکنند را نشان میدهد:

نوع داده	تعداد بایت	توضیحات	مثال
bool	۱	داده منطقی	x = true یا x = false
Boolean	۱	داده منطقی	x = true یا x = false
char	۲	داده کارکتری	x = 'A' یا x = '9'
Char	۲	داده کارکتری	x = 'A' یا x = '9'
string	بستگی به طول رشته دارد	داده رشته ای	x = "Mehdi Sobhkhiz"
String	بستگی به طول رشته دارد	داده رشته ای	x = "Mehdi Sobhkhiz"
DateTime	بستگی به طول رشته دارد	داده کار با تاریخ و زمان	x = "06/23/2009 12:00:00 pm"

نوع داده bool و Boolean نوع داده منطقی هستند که دارای دو ارزش درستی (true) یا نادرستی (false) میباشند.
نوع داده char و Char نوع داده کارکتری هستند که داده های کارکتری بین دو تا تک کوتیشن قرار میگیرند. مثل 'A'
نوع داده string و String نوع داده رشته ای هستند که داده های رشته ای بین دو تا جفت کوتیشن قرار میگرند. مثل "C# is good language"
نوع داده DateTime یک نوع داده برای کار با تاریخ و زمان است. بعداً در جای مناسب به توضیح کامل این نوع داده می پردازیم.

۳– متغیرها (Variables)

متغیرها نامی برای کلمات حافظه اند که داده ها در آن قرار میگیرند و محتویات آن ممکن است در طول اجرای برنامه تغییر کند برای مراجعه به متغیرها از نامشان استفاده می شود. لذا باید برای استفاده از حافظه ، متغیرها را نامگذاری کرد. برای نامگذاری متغیرها از حروف 'a' تا 'z' و یا حروف 'A' تا 'Z' و یا اعداد 0 تا 9 و حرف '_' میتوان استفاده کرد به شرطی که حرف اول نام متغیر عدد نباشد..
برای تعریف یک متغیر میتوان به صورت زیر عمل کرد :

نوع داده نام متغیر ;

در تعریف بالا نوع داده یکی از انواع داده ای است که در قسمت قبلی در مورد آن بحث شد و نام متغیر دلخواه میباشد.
به مثالهای زیر توجه کنید :

```
int x , y ;        تعریف متغیرهای x,y از نوع صحیح
float x , y ;      تعریف متغیرهای x,y از نوع اعشاری
String x , y ;     تعریف متغیرهای x,y از نوع رشته ای
```

نکته ۱ : برای تعریف بیش از یک متغیر بایستی آنها را با کاما (,) از هم جدا کنیم.
نکته ۲ : بهتر است برای نامگذاری متغیرها از مخفف نوع آن قبل از نام متغیر استفاده شود تا در هنگام برنامه نویسی درک کاربرد متغیر ساده تر گردد. مثلاً متغیرهای زیر را در نظر بگیرید :

```
int  intX, intY;
String str1, str2;
```

۳–۱ مقدار دادن به متغیرها

با سه روش میتوان به متغیرها در برنامه مقدار داد :

۱. هنگام تعریف متغیر :

```
int x = 100;
```

۲. بعد از تعریف متغیر با دستور انتساب (=) :

```
int x;
x = 100;
```

۳. با دستورات ورودی :

```
int x;
Console.ReadLine ( x );
```

۴– ثوابت (Constants)

ثوابت مقادیری هستند که در برنامه وجود دارند ولی قابل تغییر نیستند. قانون نامگذاری برای ثوابت از قانون نامگذاری متغیرها پیروی میکند. برای تعریف ثوابت به دو روش عمل میشود : ۱. با استفاده از دستور #define ۲. با استفاده از دستور const

برای تعریف ثوابت با استفاده از دستور #define به صورت روبرو عمل میشود : #define نام ثابت مقدار

برای تعریف ثوابت با استفاده از دستور const به صورت روبرو عمل میشود : const نوع داده نام ثابت = مقدار ;

مثلاً برای تعریف ثابت Pi = 3.14 با هر دو روش بالا به صورت زیر عمل میشود :

```
#define   Pi  3.14
const  float  Pi = 3.14;
```

۵– عملگرها (Operators)

عملگرها نمادهایی هستند که اعمل خاصی را بر روی عملوندهایشان انجام میدهند. عملگرها به چند دسته طبقه بندی میشوند :

- عملگرهای محاسباتی
- عملگرهای رابطه ای
- عملگرهای منطقی
- عملگرهای ترکیبی
- عملگرهای بیتی
- عملگرهای متفرقه

۵–۱ عملگرهای محاسباتی

این عملگرها اعمال محاسباتی ریاضی را بر روی عملوندهایشان انجام میدهند. این عملگرها در جدول زیر آمده اند :

عملگر	نام	مثال
-	تفریق و منهای یکانی	x-y یا -x
+	جمع	x+y
*	ضرب	x*y
/	تقسیم	x/y
%	باقیمانده تقسیم	x%y
--	کاهش (decrement)	x-- یا --x
++	افزایش (increment)	x++ یا ++x

عملگرهای جدول بالا بسیار ساده هستند و نیاز به توضیح اضافی ندارند.

نکته ۱ : نحوه عملکرد x++ با ++x فرق دارد ، هر چند که هر دو یک واحد به مقدار x اضافه میکنند (همینطور x-- با --x فرق دارد). با یک مثال ساده تفاوت آنها را توضیح میدهیم :

مثال : عبارات زیر را در نظر بگیرید :

```
int x , y;
x = 100;
y = x++;
```

در این مثال مقدار x و y پس از اجرای سه دستور به ترتیب برابر 101 و 100 خواهد بود. چون در خط سوم y برابر x میشود و بعداً به مقدار x یک واحد اضافه میشود. اگر به جای خط سوم عبارت y = ++x نوشته شود ، چون ابتدا به مقدار x اضافه میشود و بعداً y برابر x میشود مقدار x و y هر دو برابر 101 خواهد بود.

۵–۲ عملگرهای رابطه ای

این عملگرها ارتباط بین عملوندهایشان را مشخص میکنند. نتیجه ای که این عملگرها بر میگردانند یک مقدار منطقی است (true یا false). این عملگرها در دستورات شرطی بکار میروند. این عملگرها در جدول زیر آمده اند :

عملگر	نام	مثال
>	بزرگتر	x > y
>=	بزرگتر مساوی	x > = y
<	کوچکتر	x < y
< =	کوچکتر مساوی	x < = y
= =	متساوی	x = = y
!=	نامساوی	x != y

۵–۳ عملگرهای منطقی

عملگرهای منطقی بر روی عبارات منطقی عمل میکنند. از این عملگرها برای ترکیب شرطها نیز استفاده میشود. این عملگرها در جدول زیر آمده اند :

عملگر	نام	مثال
!	نقیض (Not)	! x
&&	و (And)	x && y
||	یا (Or)	x || y

کاربرد عملگرهای بالا به صورت زیر است :

عملگر نقیض (!) : این عملگر نتیجه عملوند منطقی خود را برعکس میکند. یعنی اگر عملوند true باشد ، false برمیگرداند و اگر false باشد true برمیگرداند.

عملگر و (&&) : در صورتی نتیجه true است وقتی که همگی عملوندها true باشند. در صورتی که یکی از عملوندها false باشد نتیجه false خواهد بود.

عملگر یا (||) : در صورتی نتیجه true است که حداقل یکی از عملوندها true باشند.

به مثال جدولی زیر توجه کنید : (T مخفف true و F مخفف false برای سادگی کار نوشته شده است).

X	Y	!X	X && Y	X || Y
T	T	F	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	F

مثال : نتیجه هر یک از عبارات زیر را بر اساس true یا false تعیین کنید :

```
int  x = 100,  y = 200,  m = 300, n = 200;
Boolean  a, b, c;
a =  ( x >= y )  &&  ( m >= n ) ;          //➔  a = false
b =  ( x >= y )  ||  ( m >= n ) ;          //➔  b = true
c =  !(x >=y )  &&  ( m>=n ) ;             //➔  c = true
```

۵-۴ عملگرهای ترکیبی

از ترکیب عملگرهای محاسباتی و عملگر = مجموعه دیگری از عملگرها بدست می آیند که هم عمل محاسباتی و هم عمل انتساب را انجام میدهند. این عملگرها در جدول زیر آمده اند :

عملگر	نام	مثال	معادل
+ =	انتساب جمع	x + = y	x = x + y
- =	انتساب تفریق	x - = y	x = x - y
* =	انتساب ضرب	x * = y	x = x * y
/ =	انتساب تقسیم	x / = y	x = x / y
% =	انتساب باقیمانده تقسیم	x % = y	x = x % y

۵-۵ عملگرهای بیتی

عملگرهای بیتی برای تست کردن ، مقدار دادن یا شیفت دادن و سایر اعمال روی یک بایت عمل میکنند. این عملگرها در جدول زیر آمده اند :

عملگر	نام
&	و (And)
|	یا (Or)
^	یای انحصاری (Xor)
~	نقیض (Not)
>>	شیفت به راست
<<	شیفت به چپ

نتیجه عملگر & وقتی یک است که هر دو بیت یک باشد. نتیجه عملگر | وقتی یک است که حداقل یکی از بیتها یک باشد. نتیجه عملگر ^ وقتی یک است که یکی از بیتها صفر و دیگری یک باشد.

در مورد بکارگیری عملگرهای شیفت (<< و >>) به صورت زیر عمل میکنیم :

به هر تعداد شیفت متغیر بر ۲ تقسیم میشود ➔// ; تعداد شیفت >> متغیر

به هر تعداد شیفت متغیر بر ۲ ضرب میشود ➔// ; تعداد شیفت << متغیر

۵-۶ عملگرهای متفرقه

یک سری از عملگرها هستند که در دسته بندی خاصی نمی گنجند و ما آنها را در دسته عملگرهای متفرقه مطرح میکنیم. این عملگرها عبارتند از :

- عملگر ? - عملگر کاما (,) - عملگر sizeof

۵-۶-۱ عملگر ?

این عملگر یک عملگر شرطی است و عبارتی را ارزیابی کرده و بر اساس ارزش آن عبارت (true یا false) ، نتیجه عبارت دیگری را در یک متغیر نگهداری میکند. نحوه کاربرد آن به صورت زیر است :

متغیر = (عبارت۱) ? (عبارت۲) : (عبارت۳)

اگر عبارت۱ دارای ارزش true باشد مقدار ارزیابی شده عبارت۲ در متغیر قرار میگیرد وگرنه مقدار ارزیابی شده عبارت۳ در متغیر قرار میگیرد.

به مثال زیر توجه کنید :

```
int x , y , max;
x = 100;
y = 200;
max = (x>=y)  ?  x  :  y;                    // ➔ max = 200
```

۵-۶-۲ عملگر کاما (,)

این عملگر برای انجام چند عمل در یک دستور به کار میرود. کاربرد این عملگر را با یک مثال توضیح میدهیم :

```
int   x , y;
y = ( x=100 ,  (x*5)/3 );
```

```
int   x , y;
x = 100;
y = ( x*5 ) / 3;
```

۵-۶-۳ عملگر sizeof

این عملگر میتواند طول یک متغیر یا یک نوع داده را بر حسب تعداد بایت تعیین کند. نحوه کاربرد این عملگر به صورت زیر است :

sizeof (**متغیر** یا **نوع داده**) ;

۶- تابع تبدیل یک نوع به نوع دیگر توسط دستور Convert

از این دستور در زبان C# برای تبدیل یک نوع به نوع دیگر استفاده میشود. از این تابع در مواقعی استفاده میکنیم که یک داده را بخواهیم با یک داده دیگر که همنوع نیستند بخواهیم با همدیگر پردازش کنیم که کامپایلر اجازه چنین کاری را به ما نمیدهد ، برای رفع این مشکل از دستور Convert استفاده میکنیم. مثلاً اگر بخواهیم یک رشته عددی مثل "123" را با یک عدد مثل ۱۰۰ جمع کنیم بایستی ابتدا رشته را به عدد تبدیل کنیم و سپس با عدد جمع کنیم ، دستورات زیر را در نظر بگیرید :

```
String s1 = "123";
int x = 100;
x = x + Convert.ToInt32 ( s1 );
```

در خط سوم رشته s1 به یک داده صحیح ۴ بایتی تبدیل میشود و با x که یک داده صحیح ۴ بایتی است جمع میشود و نتیجه در x جمع میشود.

انواع تبدیلات داده ای در جدول زیر آمده است :

نوع داده	توضیحات
ToBoolean	تبدیل نوع منطقی
ToByte	تبدیل نوع بایت
ToChar	تبدیل نوع کارکتری
ToDecimal	تبدیل نوع اعشاری
ToDouble	تبدیل نوع اعشاری
ToSingle	تبدیل نوع اعشاری
ToInt16	تبدیل نوع صحیح
ToInt32	تبدیل نوع صحیح
ToInt64	تبدیل نوع صحیح
ToUInt16	تبدیل نوع صحیح بدون علامت
ToUInt32	تبدیل نوع صحیح بدون علامت
ToUInt64	تبدیل نوع صحیح بدون علامت
ToString	تبدیل رشته ای

یک نکته در مورد عملگر + : از این عملگر برای جمع دو رشته نیز استفاده میشود.

مثال : در قطعه کد زیر دو داده که یکی رشته ای و دیگری صحیح می باشد با یکدیگر جمع رشته ای می شوند و نتیجه در داده رشته ای ذخیره میشود :

```
int  x = 3260712;
String  s1 = "My phone number is : ";
s1  =  s1  +  x.ToString( );          ➔      s1 = "My phone number is 3260712"
```

۷– نوشتن برنامه ها در کنسول (Console)

در #C محیطی برای نوشتن برنامه های تحت غیر ویندوزی وجود دارد که از آن برای حل بعضی از برنامه ها مثل چند نخی (multi threading) و ... استفاده می شود. برای ایجاد یک برنامه کنسولی در سی شارپ مراحل زیر را دنبال کنید :

از منوی File گزینه New و سپس Project را کلیک کنید تا کادر New Project باز شود. این کادر دارای سه قسمت زیر است :

- Project Types : این قسمت نوع زبان پروژه و محیط پروژه را مشخص میکند ، از این قسمت Visual C# و در قسمت زیرین آن Windows را انتخاب کنید.
- Templates : نوع پروژه ای که میخواهیم بنویسیم در این قسمت مشخص میشود. انواع پروژه مثل کنسول ، ویندوزی و .... از این قسمت Console Application را انتخاب کنید.
- Project Name : در این قسمت نام پروژه و محل ذخیره آن مشخص میشود. در قسمت Name ، نام پروژه را بنویسید و در قسمت Location ، مسیر ذخیره پروژه را بنویسید.

۷–۱ برخی توابع مهم مربوط به کلاس کنسول

در این قسمت به معرفی برخی از توابع مهم مربوط به کلاس کنسول می پردازیم. این توابع در جدول زیر شرح داده میشوند :

تابع	شرح کار تابع	مثال
Clear	این تابع صفحه نمایش را پاک می کند	`Console.Clear();`
ReadKey	این تابع برای خواندن هر کلیدی از صفحه کلید استفاده میشود	`Console.ReadKey();`
ReadLine	این تابع برای گرفتن ورودی از صفحه کلید استفاده میشود.	`int x;` `x = Convert.ToInt32(Console.ReadLine());`
Write	این تابع برای نوشتن داده در خروجی صفحه نمایش استفاده میشود	`Console.Write (x.ToString());`
WriteLine	این تابع برای نوشتن داده در خروجی صفحه نمایش استفاده میشود (در انتها به سطر بعدی می رود)	`Console.WriteLine(x.ToString());`

۷–۲ آرگومانهای جایگزین در چاپ اطلاعات با استفاده از {}

برای چاپ اقلام (بیش از یک قلم داده) در خروجی با استفاده از توابع Write و WriteLine از آرگومانهای جایگزین بایستی استفاده کرد. طرز استفاده از این آرگومانها را با مثالی ذکر میکنیم :

مثال : برنامه ای بنویسید که طول و عرض مستطیلی را از ورودی بخواند و مساحت و محیط آن را محاسبه کرده و در خروجی با پیغامهای مناسبی نمایش دهد.

```
int x, y;
int area, perim;
Console.Write("Enter width of rectangle : ");
x = Convert.ToInt32(Console.ReadLine());
Console.Write("Enter height of rectangle : ");
y = Convert.ToInt32(Console.ReadLine());
area = x * y;
perim = 2 * (x + y);
Console.WriteLine("Area is {0} , Perim is {1}",area,perim);          (*)
Console.Write("\n\nPress any key to exit");
Console.ReadKey();
```

در دستوری که زیر آن خط کشیده شده است (*) از آرگومانهای جایگزین 0 و 1 درون {} استفاده شده است. این بدین معنی است که به جای {0} مقدار متغیر area و به جای {1} مقدار متغیر perim قرار میگیرد. کارکرد این آرگومان جایگزین بسیار شبیه به ٪ در زبان C می‌باشد.

۸- مثال در کلاس

۸-۱. برنامه ای بنویسید که سه عدد صحیح از ورودی بخواند و میانگین آنها را در خروجی نمایش دهد :

```
int x, y, z;
double ave;
Console.Write("Enter x : ");
x = Convert.ToInt32(Console.ReadLine());
Console.Write("Enter y : ");
y = Convert.ToInt32(Console.ReadLine());
Console.Write("Enter z : ");
z = Convert.ToInt32(Console.ReadLine());
ave = (x + y + z) / 3.0;
Console.WriteLine("Average is {0} ", ave);
Console.Write("\n\nPress any key to exit");
Console.ReadKey();
```

در دستوری که زیر آن خط کشیده شده از 3.0 به جای 3 برای تقسیم اعشاری استفاده کردیم. این کار برای حفظ مقدار اعشار می باشد.

۸-۲. برنامه ای بنویسید که دو مقدار از ورودی بخواند و محتویات آنها را با استفاده از متغیر کمکی تعویض کند :

```
int x, y;
int temp;
Console.Write("Enter x : ");
x = Convert.ToInt32(Console.ReadLine());
Console.Write("Enter y : ");
y = Convert.ToInt32(Console.ReadLine());
temp = x;
x = y;
y = temp;
Console.WriteLine("x is {0}  ,  y is {1} ", x, y);
Console.Write("\n\nPress any key to exit");
Console.ReadKey();
```

۹- تمرین در منزل

۹-۱. هر یک از عبارات زیر را به عبارت #C تبدیل کنید.

1. $y = p \times r^2 + \frac{w}{x} + p$

2. $y = k^2 \times m^3 \times \frac{n}{p}$

۹-۲. برنامه ای بنویسید که شعاع دایره ای را که یک مقدار اعشاری است از ورودی بخواند و مساحت و محیط آن را در خروجی نمایش دهد :

مساحت = (شعاع * شعاع) * 3.14

محیط = 2 * شعاع * 3.14

۹-۳. برنامه ای بنویسید که سه مقدار صحیح از ورودی بخواند و ماکزیمم آنها را در خروجی چاپ کند : (از عملگر ? استفاده کنید)

۹-۴. برنامه ای بنویسید که اصل پول ، نرخ بهره ، و روزها را از ورودی دریافت کرده ، سود آن را به صورت زیر محاسبه و سپس چاپ کند.

سود = اصل پول × نرخ بهره × روزها / 365

۹-۵. برنامه ای بنویسید که معدل ۵ نمره درسی یک دانشجو را محاسبه کند (هر درس دارای واحد خاص خودش می باشد)

فصل دوم : توانایی کار با محیط سی شارپ

۱- نوار منو (menu bar)
۲- نوار ابزار (tool bar)
۳- جعبه ابزار (tool box)
۴- پنجره خواص (property Box)
۵- پنجره جستجوگر فایلها (Solution Explorer)
۶- طرز ایجاد یک پروژه
۷- اضافه کردن کنترل ها به برنامه (drag and drop)
۸- معرفی برخی از کنترل ها که زیاد مورد استفاده قرار میگیرند (Button – textbox – Label - Form).
۹- معرفی برخی خواص مهم کنترلها (Properties)
۱۰-معرفی برخی رویدادهای کنترلها (Events)
۱۱-معرفی برخی متدهای کنترل (Methods)
۱۲- مقدار دادن به خاصیت یک کنترل در هنگام کدنویسی با استفاده از نقطه + خاصیت های تک مقداری و چند مقداری
۱۳-مثال در کلاس
۱۴-تمرین در خانه

برای آنکه بتوانیم راحت‌تر با Visual C#.Net برنامه بنویسیم ، بایستی با محیط برنامه‌نویسی آن آشنا شده و با امکانات قرار داده شده در این محیط راحت ارتباط برقرار کنیم. در این فصل ما به اختصار برخی از امکانات مهم از قبیل جعبه ابزار ، نوار ابزار ، نوار منو ، جعبه خواص و غیره را مورد بررسی قرار می‌دهیم.

۲,۱. **نوار منو** (Menu Bar) **:** شکل نوار منو در زیر نشان داده شده است.

از این نوار برای دسترسی به امکانات قرار داده شده در سی‌شارپ استفاده میشود. این نوار دارای ۱۲ منوی زیر میباشد :

۱. منوی File : این منو با پروژه برنامه سروکار دارد ، برخی از گزینه های آن عبارتند از .
   - New : از این گزینه برای ایجاد پروژه جدید استفاده می‌شود.
   - Open : این گزینه برای باز کردن پروژه ای که قبلاً ذخیره کردیم استفاده می‌شود.
   - Add : این گزینه برای اضافه کردن یک پروزه به پروژه جاری استفاده می‌شود.
   - Close : این گزینه پنجره جاری که در محیط سی‌شارپ باز است را می‌بندد.
   - Close Solution : این گزینه پروژه را می‌بندد.
   - Save : این گزینه پنجره جاری را ذخیره می‌کند.
   - Save As : با این گزینه می‌توان پنجره جاری را با نام دیگری نیز ذخیره نمود.
   - Save All : این گزینه کل پنجره‌های یک پروژه و به اضافه خود پروژه را ذخیره می‌کند.
   - Exit : برای خروج از محیط سی‌شارپ از این گزینه استفاده می‌شود.

۲. منوی Edit : این منو بسیار ساده است و بشتر برنامه‌های مختلف دارای این منو می‌باشند که طرز کار آن در برنامه‌ها یکسان است و ما از بررسی این منو در اینجا صرفنظر می‌کنیم.

۳. منوی View : از این منو برای مخفی یا آشکار کردن برخی از پنجره‌ها و جعبه‌ها در برنامه استفاده می‌شود.

۴. منوی Project : از این منو برای اضافه کردن فرم ، اضافه کردن کنترل کاربر ، اضافه کردن آیتم جدید یا آیتم موجود ، ... به پروژه استفاده می‌شود.

۵. منوی Build : از این منو برای کامپایل یا ترجمه برنامه استفاده می‌شود. (برنامه قبل از اجرا بایستی به زبان ماشین ترجمه شود).

۶. منوی Debug : از این برنامه برای اجرای برنامه ، توقف برنامه و تریس کردن برنامه استفاده می‌شود.

۷. منوی Data : از این منو برای دسترسی به داده منبع و کوئری استفاده می‌شود.

۸. منوی Format : از این منو برای تنظیم کردن (از قبیل سایز ، موقعیت و ...) کنترل قرار داده شده در فرم برنامه استفاده می‌شود.

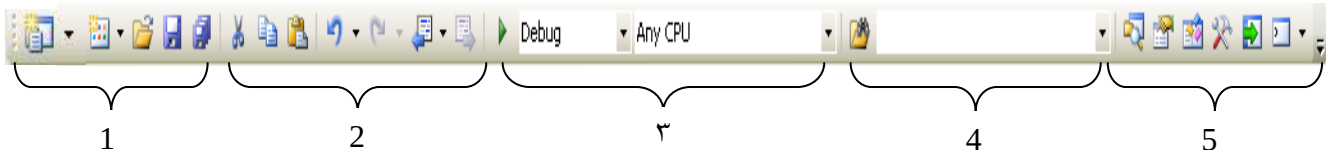
۹. منوی Tools : در این منو برخی از امکانات حرفه‌ای سی‌شارپ از قبیل متصل شدن به بانک اطلاعاتی (Connect to database) ، تنظیم محیط سی‌شارپ (Options) و ... قرار داده شده است.

۱۰. Test : از این منو برای تست برنامه استفاده می‌شود.

۱۱. Window : این منو برای طرز نمایش پنجره‌های باز موجود در برنامه استفاده می‌شود.

۱۲. Help : از این منوی برای دسترسی به Help (اگر MSDN نصب باشد) سی‌شارپ استفاده می‌شود.

۲,۲. **نوار ابزار** (ToolBar) **:** از این نوار برای دسترسی سریعتر به برخی از امکانات سی‌شارپ استفاده می‌شود. شکل این نوار در زیر آمده است :



این نوار را به ۵ قسمت بالا تقسیم کردیم :

قسمت 1 : کار برخی از گزینه‌های منوی File را انجام می‌دهد.

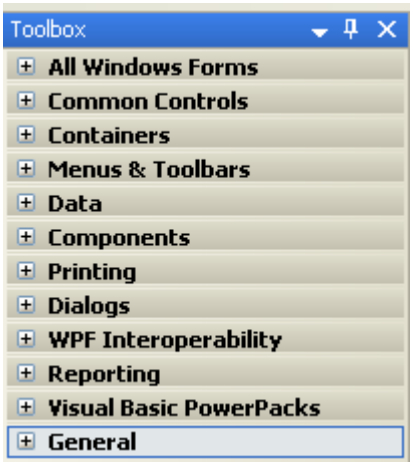
قسمت 2 : کار برخی از گزینه منوی Edit را انجام می‌دهد.

قسمت 3 : اجرا و نحوه اجرای برنامه را مشخص می‌کند.

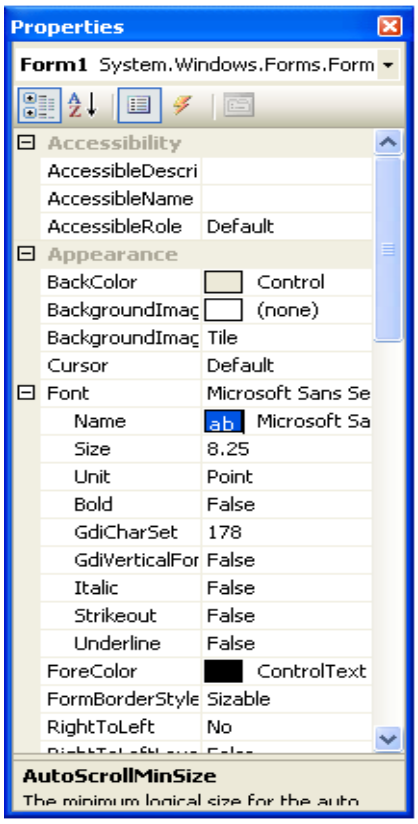
قسمت ۴: برای جستجوی یک کلمه در پروژه استفاده می‌شود.

قسمت ۵ : کار برخی از گزینه‌های منوی View را برای آشکار کردن پنجره‌ها انجام می‌دهد.

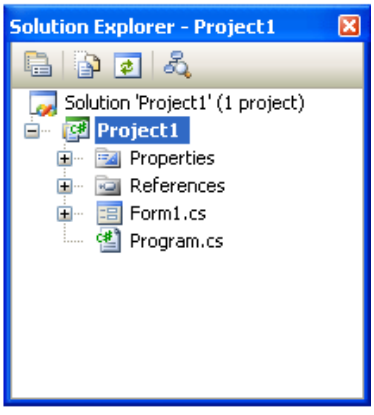
۲،۳. **جعبه ابزار** (ToolBox) **:** از این جعبه برای دسترسی به کنترل‌ها و اضافه کردن آنها به فرم برنامه استفاده می‌شود. شکل آن به صورت زیر است:



۲،۴. **جعبه خواص** (Property window) **:** از این جعبه برای تنظیم خواص و رویدادهای کنترلهای موجود در فرم برنامه استفاده می‌شود. شکل آن به صورت زیر است :



۲،۵. **پنجره جستجوگر فایلهای پروژه** (Solution Explorer) **:** از این پنجره برای کار با فرمها و کدهای موجود در برنامه استفاده می‌شود. این پنجره در شکل زیر نمایش داده شده است.



۲,۶. **طرز ایجاد یک پروژه جدید** (Windows Application) **:** برای ایجاد یک پروژه جدید بایستی از منوی File گزینه New و سپس Project را انتخاب کنید. با این کار کادری باز می‌شود. که مثل طرز ایجاد برنامه کنسولی می‌باشد ولی در اینجا در قسمت Templates گزینه Windows Forms Application را انتخاب کنید و نام پروژه و مسیر ذخیره آن را بنویسید و دکمه OK را بزنید.

۲,۷. **اضافه کردن کنترل ها به فرم برنامه** : از درون جعبه ابزار (ToolBox) کنترلی که میخواهیم به فرم برنامه اضافه کنید انتخاب و آن را روی فرم برنامه بکشید.

۲,۸. **معرفی برخی از کنترل ها که زیاد مورد استفاده قرار می‌گیرند :** در این قسمت به بررسی برخی کنترل بسیار مهم که زیاد در برنامه نویسی مورد استفاده قرار می‌گیرند می‌پردازیم. این کنترل‌ها عبارتند از : فرم – برچسب – جعبه متن – دکمه.

- **فرم** (Form) **:** فرم برنامه برای نگهداری کنترلهای مورد نیاز در برنامه استفاده می‌شود.
- **برچسب** (Label) **:** از این کنترل برای نمایش متن درون فرم بکار برده می‌شود.
- **جعبه متن** (TextBox) **:** از این کنترل برای دریافت متن یا مقدار از ورودی استفاده می‌شود.
- **دکمه** (Button) **:** از این کنترل برای اجرای یک رویداد یا تابع برنامه استفاده می‌شود.

۲,۹. **معرفی برخی خواص مهم کنترلها** (Properties) **:** هر کنترل یا هر شی دارای خاصیت‌های خاص خودش می‌باشد، در این قسمت به برخی از خواص مهم کنترل‌های تعریف شده می‌پردازیم. برخی از خواص که تعریف می‌کنیم مشترک هستند.

برخی از خواص مربوط به فرم (Form)

خاصیت	توضیحات
Name	نام فرمی را که داریم با آن کار می‌کنیم مشخص میکند. از این خاصیت در هنگام کدنویسی زیاد استفاده می‌شود.
Text	عنوان فرم یا متنی که در بالای فرم نوشته شده است را مشخص می‌کند.
AcceptButton	دکمه‌ای در فرم را مشخص میکند که اگر ما در فرم کلید Enter را بزنیم آن دکمه اجرا شود.
BackColor	رنگ پس زمینه فرم را تعیین می‌کند.
BackgroundImage	عکس پس زمینه فرم را تعیین می‌کند.
BackgroundImageLayout	طرز نمایش عکس پس زمینه فرم را مشخص می‌کند.
CancleButton	دکمه‌ای در فرم را مشخص میکند که اگر ما در فرم کلید ESC را بزنیم آن دکمه اجرا شود.
Enabled	این خاصیت دو مقدار True و False دارد. اگر False باشد فرم غیرفعال و اگر True باشد فرم فعال است.
Font	این خاصیت تنظیمات فونت فرم را انجام می‌دهد.
ForeColor	این خاصیت رنگ متن فرم را تعیین می‌کند.
FormBorderStyle	این خاصیت سبک حاشیه های فرم را تعیین می‌کند.
Icon	این خاصیت آیکونی را که در بالا و قسمت چپ فرم قرار دارد مشخص می‌کند.
IsMdiContainer	این خاصیت تعیین می‌کند که آیا این فرم ، فرم‌های دیگر را درون خود جای دهد یا خیر؟ True یا False
MaximumBox	این خاصیت تعیین میکند آیا دکمه ماکزیمم مربوط به فرم نمایش داده شود یا خیر؟ True یا False
MinimizeBox	این خاصیت تعیین میکند آیا دکمه مینیم مربوط به فرم نمایش داده شود یا خیر؟ True یا False
Opacity	این خاصیت شفافیت فرم را به درصد نشان می‌دهد. در کدنویسی بایستی این مقدار اعشاری باشد. (بین صفر تا یک)
RightToLeft	این خاصیت نشان می‌دهد که متن‌ها از چپ با راست نمایش داده شوند (False) یا از راست به چپ (True).
ShowIcon	این خاصیت نشان می‌دهد که آیکون بالای فرم نشان داده شود یا خیر؟ True یا False
Size	این خاصیت اندازه فرم را تعیین می‌کند. که این خاصیت دارای دو قسمت width یعنی طول و height یعنی ارتفاع فرم می‌باشد.
StartPosition	این خاصیت نشان می‌دهد که فرم در هنگام اجرای برنامه در کجای صفحه نمایش ظاهر شود.
WindowState	این خاصیت نشان می‌دهد که فرم در هنگام اجرا به صورت عادی ، ماکزیمم و یا مینیمم ظاهر شود.

نکته : تعدادی از خاصیت‌هایی که برای هر یک از کنترل‌ها گفته می‌شود ، برای دیگر کنترل‌ها نیز وجود دارد و همان وظایف را انجام می‌دهد. مانند Name ، Text و BackColor و غیره.

برخی از خواص مربوط به برچسب (Label)

خاصیت	توضیحات
AutoSize	این خاصیت یک مقدار منطقی است که اگر True باشد اندازه کنترل با متن نوشته شده تنظیم می‌شود.
Location	این خاصیت مکان کنترل را نسبت به فرم تعیین می‌کند.
Image	عکسی را برای برچسب انتخاب می‌کند
ImageAlign	مکان عکس را برای برچسب تعیین می‌کند
Modifiers	این خاصیت اگر Private باشد کنترل فقط در همان فرم قابل استفاده است و اگر Public باشد در تمامی فرم‌ها استفاده می‌شود.
Text	عنوان متن برچسب را عوض می‌کند
TextAlign	مکان متن را تنظیم می‌کند
Visible	اگر این خاصیت False شود دیگر در هنگام اجرای برنامه کنترل مورد نظر مخفی می‌شود

برخی از خواص مربوط به جعبه متن (TextBox)

خاصیت	توضیحات
BorderStyle	حاشیه جعبه متن را تغییر میدهد
MaxLength	حداکثر تعداد کارکترهایی که میتوان درون جعبه متن نوشت را نشان می‌دهد
MultiLine	اگر این خاصیت True شود درون جعبه متن چندین خط میتوان تایپ کرد.
PasswordChar	این خاصیت برای رمزی کردن جعبه متن بکار می‌رود. اگر هر کارکتری درون این خاصیت قرار داده شود. محتویات متن نوشته شده به صورت رمزی (با همان کارکتر) نمایش داده می‌شوند.
ReadOnly	اگر این خاصیت True شود دیگر در هنگام اجرای برنامه به صورت دستی نمی‌توان متن جعبه متن را تغییر داد (ولی از طریق کدنویسی می‌شود این کار را انجام داد)
Text	متن جعبه متن را تغییر می‌دهد
TextAlign	مکان متن را درون جعبه متن تنظیم می‌کند

برخی از خواص مربوط به دکمه (Button)

خاصیت	توضیحات
BackgroundImage	تصویر پس زمینه دکمه را میتوان با این خاصیت عوض کرد
BackgroundImageLayout	نوع نمایش عکس پس زمینه را مشخص می‌کند
ForeColor	رنگ متن نوشته شده در دکمه را تغییر می‌دهد
BackColor	رنگ پس زمینه دکمه را تغییر می‌دهد
Text	متن دکمه را تغییر می‌دهد
Enabled	اگر این خاصیت False شود دکمه غیر فعال می‌شود
Visible	اگر این خاصیت False شود دکمه مخفی می‌شود

۲,۱۰. **معرفی برخی رویدادهای مهم کنترلها (Events) :** هر کنترل یا هر شی دارای رویدادهای خاص خودش می‌باشد ، در این قسمت به بررسی برخی از رویدادهای مهم مربوط به کنترل‌ها می‌پردازیم.

برخی از رویدادهای مربوط به فرم و کنترل‌ها

رویداد	توضیحات
Click	وقتی کنترل کلیک می‌شود این رویداد رخ می‌دهد
DoubleClick	وقتی کنترل دابل‌کلیک می‌شود این رویداد رخ می‌دهد
FormClosed	وقتی فرم بسته شود این رویداد رخ می‌دهد
FormClosing	وقتی فرم در حال بسته شدن باشد این رویداد رخ می‌دهد
KeyDown	وقتی کلیدی درون کنترل فشرده شود این رویداد رخ می‌دهد
KeyPress	وقتی کلیدی درون کنترل فشرده شود این رویداد رخ می‌دهد

KeyUp	وقتی کلید فشرده شده رها شود این رویداد رخ می‌دهد
MouseClick	وقتی روی فرم کلیک ماوس صورت گیرد (چه چپ کلیک و چه راست کلیک) این رویداد رخ می‌دهد
MouseDown	وقتی کلید ماوس فشرده شود این رویداد رخ می‌دهد
MouseMove	وقتی ماوس روی کنترل برود این رویداد رخ می‌دهد
MouseLeave	وقتی ماوس کنترل را ترک کند این رویداد رخ می‌دهد
MouseUp	وقتی کلیک ماوس رها شود این رویداد رخ می‌دهد
Load	وقتی فرم لود می‌شود این رویداد رخ می‌دهد
Resize	وقتی اندازه فرم تغییر می‌کند این رویداد رخ می‌دهد
TextChanged	وقتی متن درون برچسب تغییر کند این رویداد رخ می‌دهد

۲,۱۱. **معرفی برخی متدهای مهم کنترلها** (Methods) **:** هر یک از کنترل‌ها دارای متدهای خاص خودشان می‌باشند ، به ترتیب برای هر یک از کنترل‌ها این متدها را بررسی کنیم. (از این متدها فقط در هنگام کدنویسی استفاده می‌شود).

برخی از متدهای مربوط به فرم (Form)

متد	توضیحات
Close( )	این متد باعث بسته شدن فرم می‌شود
Hide( )	این متد باعث مخفی کردن فرم می‌شود
Scale(factor)	این متد باعث تغییر اندازه فرم می‌شود. یک مقدار صحیح هم به عنوان ورودی می‌گیرد. (مثلا اگر مقدار ۲ وارد کنیم فرم دوبرابر می‌شود)
Show( )	این متد باعث ظاهر شدن فرم می‌شود
ShowDialog( )	این متد باعث ظاهر شدن فرم می‌شود – فرق این متد با متد قبلی این است که تا زمانی که فرم باز است ، به فرم زیرین دسترسی نداریم

برخی از متدهای مربوط به جعبه متن (TextBox)

متد	توضیحات
AppendText (متن)	این متد ، متنی را به متن موجود در جعبه متن اضافه می‌کند.
Clear( )	این متد ، متن موجود در جعبه متن را پاک می‌کند.
Copy( )	متن انتخاب شده درون جعبه متن را درون clipboard ذخیره می‌کند
Cut( )	متن انتخاب شده را از درون جعبه متن حذف و آنرا درون clipboard ذخیره می‌کند
Paste( )	متن ذخیره شده درون clipboard را به داخل جعبه متن قرار می‌دهد
SelectAll()	تمامی متن درون جعبه متن را انتخاب می‌کند
Focus( )	این متد اشاره گر صفحه کلید را به درون کنترل مورد نظر می‌فرستد
Undo( )	عملیات قبلی مربوط به جعبه متن را لغو می‌کند

۲,۱۲. مقدار دادن به خاصیت یک کنترل در هنگام کدنویسی : برای اینکه بتوانیم به خاصیت یک کنترل در هنگام کدنویسی مقدار دهیم از علامت نقطه استفاده می‌کنیم. طرز استفاده آن به صورت زیر است :

```
نام‌کنترل . خاصیت = مقدار ;
```

مثال. فرض کنید جعبه متنی داریم به نام txtName که می‌خواهیم متن آن به "Hello" تغییر دهیم ، به صورت زیر عمل می‌کنیم :

```
txtName.Text = "Hello";
```

خاصیت‌ها به دو دسته تقسیم می‌شوند : ۱- خاصیت های تک مقداری مانند مثال بالا ۲- خاصیت‌های چند مقداری

خاصیت‌های چند مقداری را درون جعبه خواص به صورت کشویی ملاحظه می‌کنیم. برای مقدار دادن به این خاصیت‌ها هنگام کد نویسی بایستی از مقداری که درون این خاصیت‌های کشویی قرار دارد استفاده کرد. مثلا اگر بخواهیم جعبه متن مثال بالا را به صورت فقط خواندنی در بیاوریم به صورت زیر عمل می‌کنیم :

```
txtName.ReadOnly = false;
```

یا اگر بخواهیم متن نوشته شده درون جعبه متن را در مرکز جعبه متن نمایش دهیم :

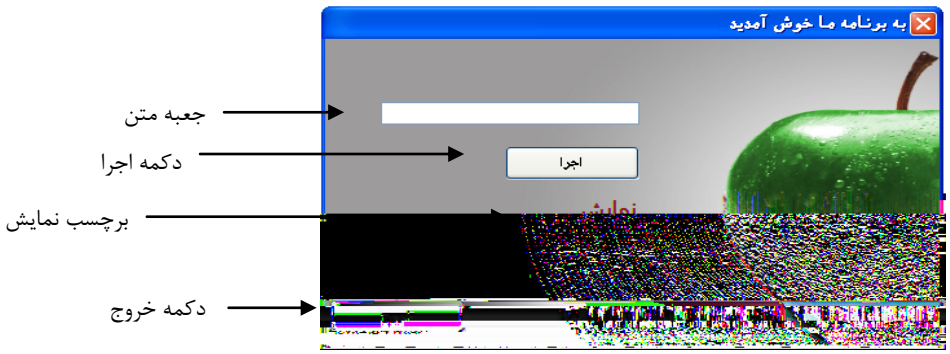
```
txtName.TextAlign = HorizontalAlignment.Center;
```

۲,۱۳. مثال در کلاس

۱- **برنامه ای بنویسید که دارای یک فرم ، یک برچسب ، یک جعبه متن و دو دکمه به مشخصات زیر داشته باشد :**

کنترل	خاصیت	مقدار
Form	Name	myForm
	BackColor	SkyBlue
	BackGroundImage	یک عکس دلخواه
	BackGroundImageLayout	Stretch
	FormBorderStyle	FixedDialog
	MaximumBox	False
	MinimizeBox	False
	RightToLeft	Yes
	StartPosition	CenterScreen
	Text	به برنامه ما خوش آمدید
Label	Name	Label1
	BackColor	Transparent
	FontName	Microsoft Sans Serif
	FontStyle	Bold
	FontSize	14
	ForeColor	ForestGreen
	Text	نمایش
Button1	Name	btnRun
	Text	اجرا
TextBox	Name	textBox1
	Text	خالی
	Size	200;20
Button2	Name	btnExit
	Text	خروج

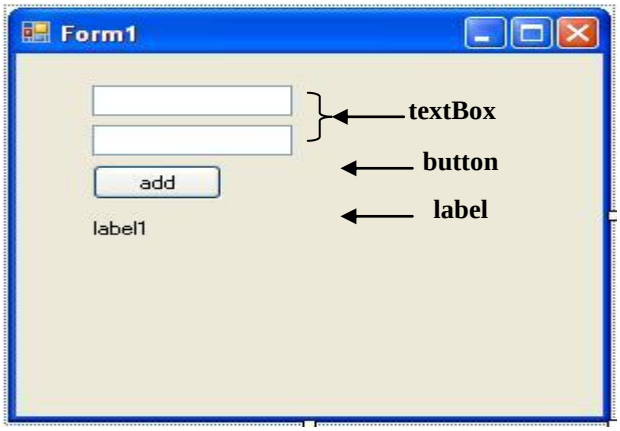
این فرم را به صورت زیر با مشخصات داده شده بالا به صورت زیر طراحی کنید :



حال میخواهیم با این فرم و کنترلهای نصب شده ، برنامه ای بنویسیم که اگر کاربر درون جعبه متن هر چیزی وارد کرد ، با زدن دکمه "اجرا" متن درون برچسب نمایش داده شود. با زدن دکمه خروج نیز فرم بسته شود.

```
private void btnRun_Click(object sender, EventArgs e)
{
   label1.Text = textBox1.Text;
}
private void button1_Click(object sender, EventArgs e)
{
   Close();
}
```

2- برنامه ای بنویسید که فرم آن به شکل زیر باشد و دو ورودی که اعداد صحیح هستند از ورودی دریافت کند و با زدن دکمه add مجموع دو عدد ورودی را درون برچسب نمایش دهد



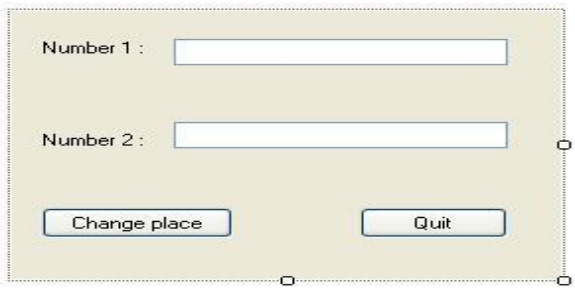
روی دکمه add دابل کلیک کنید و کد زیر را درون آن بنویسید :

```
private void button1_Click(object sender, EventArgs e)
{
    int x, y;
    x = Convert.ToInt32(textBox1.Text);
    y = Convert.ToInt32(textBox2.Text);
    label1.Text = (x + y).ToString();
}
```

۳- برنامه‌ای بنویسید که شعاع دایره را از ورودی بگیرد و مساحت و محیط آن را با زدن دکمه‌ای نمایش دهد.


```
private void btnCalc_Click(object sender, EventArgs e)
{
  //برنامه محاسبه مساحت
  double Area, Perim;
  double pi = Math.Atan(1) * 4;
  double radius = Convert.ToDouble(textBox1.Text);
  Area = pi * (radius * radius);
  Perim = 2 * pi * radius;
  label1.Text = Area.ToString();
  label2.Text = Perim.ToString();
}
```

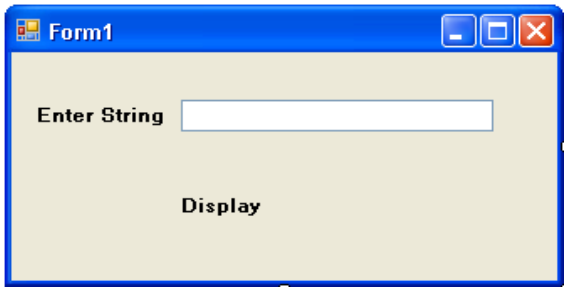
۴- برنامه ای بنویسید که دو مقدار از دو textBox بگیرد و جای مقدار دو textBox را با زدن دکمه‌ای عوض کند ، یک دکمه برای خروج بگذارید. ظاهر فرم را به شکل زیر طراحی کنید.



```
private void button1_Click(object sender, EventArgs e)
{
      int num1 = Convert.ToInt32(textBox1.Text);
      int num2 = Convert.ToInt32(textBox2.Text);
      int temp;
      temp = num1;
      num1 = num2;
      num2 = temp;
      textBox1.Text = num1.ToString();
      textBox2.Text = num2.ToString();
}

private void button2_Click(object sender, EventArgs e)
{
    Form1.ActiveForm.Close();   یا   Close();     یا     this.Close();
}
```

5- **برنامه‌ای بنویسید که از ورودی یک رشته را دریافت می‌کند آن را همزمان درون برچسب** Display **نمایش دهد. از رویداد** KeyUp **مربوط به جعبه متن استفاده کنید.**




```
private void textBox1_KeyUp(object sender, KeyEventArgs e)
{
      label2.Text = textBox1.Text;
}
```

۶- **برنامه‌ای بنویسید که از ورودی یک رشته را دریافت می‌کند همزمان که کلیدهای صفحه کلید را برای تایپ متن وارد می‌کنیم آخرین کارکتر تایپ شده درون برچسب نمایش دهد.**

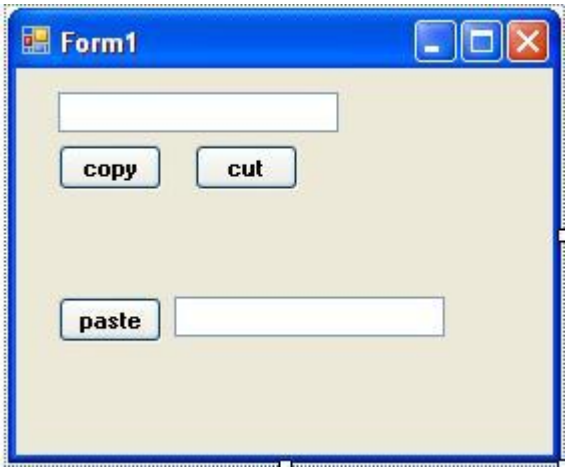

```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
      label2.Text = e.KeyChar.ToString();
}
```

نکته۱ : Sender مشخص کننده شیئی است که رویداد را ارسال میکند و e مربوط به کلید فشرده شده است. خط سوم معادل کارکتری کلید فشرده شده را به رشته تبدیل میکند و در label2 قرار میدهد.

نکته۲ : KeyDown و KeyUp **برخلاف** KeyPress تمامی کلیدهای صفحه کلید را میتوانند بررسی کنند ، اما بین حروف کوچک و بزرگ انگلیسی تفاوتی قائل نمی‌شوند (یعنی برای حروف بزرگ و کوچک انگلیسی یک کد اسکی در نظر می‌گیرد).

نکته۲ : هر کارکتر درون صفحه کلید برای خودش یک کد عددی دارد که به این کد عددی ، کد اسکی (Ascii Code) گفته می‌شود.

۷- **برنامه‌ای بنویسید که فرم آن به صورت زیر باشد و کارهای** copy ، cut **و** paste **را انجام دهد.**




```
private void button1_Click(object sender, EventArgs e)
{
     textBox1.Copy();
}

private void button2_Click(object sender, EventArgs e)
{
     textBox1.Cut();
}

private void button3_Click(object sender, EventArgs e)
{
     textBox2.Clear();
     textBox2.Paste();
}
```

طریقه اجرای برنامه بدین صورت است که ابتدا متنی را داخل جعبه متن مینویسیم بعد قسمتی یا تمام متن داخل جعبه متن را با ماوس یا صفحه کلید انتخاب میکنیم و سپس دکمه Cut یا Copy را میزنیم تا متن انتخاب شده وارد حافظه ClipBord شود. بعد با زدن دکمه Paste متنی که در حافظه ClipBoard قرار دارد را به داخل جعبه متن دوم قرار داده میشود.

۲,۱۴. تمرین در منزل

۱- **برنامه ای بنویسید که طول و عرض مستطیلی را از ورودی بگیرد ، با زدن دکمه اول مساحت را درون برچسب اول و با زدن دکمه دوم محیط مستطیل را درون برچسب دوم نمایش دهد.**

۲- **برنامه‌ای که سن یک فرد را بر حسب روز از ورودی خوانده و در خروجی سن فرد را بر اساس سال و ماه و هفته و روز نمایش دهد.**

۳- **فرق بین رویدادهای KeyDown و KeyPress چیست ؟ در مورد آن تحقیق کنید و یک مثال از هر کدام حل کنید تا فرق بین این دو رویداد را متوجه شوید.**

۴- **برنامه‌ای بنویسید که دو عدد صحیح از ورودی خوانده محتوای آن دو را با هم عوض کند (بدون استفاده از متغیر کمکی).**

۵- **مثال ۶ را به گونه‌ای تغییر دهید که به جای آخرین کارکتر فشرده شده ، کد اسکی آخرین کارکتر فشرده شده نمایش داده شود.**

فصل سوم : ساختارهای تصمیم

۱- ساختارهای تصمیم
۲- ساختار تصمیم گیری if و if else
۳- عملگرهای رابطه‌ای و منطقی در دستورات شرطی
۴- معرفی کنترلهای دکمه رادیویی (RadioButton) و جعبه انتخاب (CheckBox)
۵- ساختار تصمیم switch
۶- معرفی کنترلهای جعبه لیست (ListBox) و جعبه کشویی (ComboBox)
۷- معرفی جعبه پیغام (MessageBox)
۸- تمرین در خانه

ساختارهای تصمیم و تکرار

در حالت عادی ، دستورات برنامه از اولین دستور به آخرین دستور اجرا میشوند. اگر بخواهیم بعضی از دستورات چندین بار اجرا شوند و بعضی از دستورات تحت شرایط خاصی اجرا شوند و بعضی دیگر اجرا نشوند از ساختارهای تصمیم و تکرار استفاده میکنیم. برنامه هایی که در فصلهای قبلی نوشتیم فاقد هرگونه تصمیم گیری و تکرار بودند. برنامه های مهم و پیچیده بدون استفاده از ساختار تصمیم گیری و حلقه های تکرار قابل پیاده سازی نیستند. به همین دلیل در این فصل ساختارهای تصمیم گیری و در فصل بعدی ساختارهای تکرار را بررسی میکنیم و همراه آنها کنترل های بیشتری و مثالهای بیشتری را معرفی می‌کنیم.

۱-.ساختارهای تصمیم :

اگر بخواهیم تحت شرایطی ، تعدادی از دستورات اجرا شوند و یا تعدادی دیگر از دستورات اجرا نشوند ، باید از ساختارهای تصمیم استفاده کنیم. این ساختارها شرطی را تست کرده و در صورت درست بودن شرط ، مجموعه ای از دستورات اجرا میشوند. ساختارهای تصمیم که در C#.Net وجود دارند. عبارتند از if و switch .

۲- ساختار تصمیم if :

این ساختار شرطی را تست می‌کند و در صورتی که آن شرط دارای ارزش درستی باشد. مجموعه ای از دستورات را اجرا میکند کاربرد دستور if به صورت زیر است :

کاربرد اول(تک دستوری)	کاربرد دوم (چند دستوری)

if (شرط)
دستور۱ ;
else
دستور۲;
	if (شرط)
	{
	دستور۱ ;
	دستور۲ ;
	دستورn ;
	}
	else
	{
	دستور e1 ;
	دستور e2 ;
	دستور en ;
	}

در کاربرد اول چنانچه شرط دارای ارزش درستی باشد ، دستور ۱ اجرا میشود در غیر اینصورت دستور۲. در کاربرد دوم نیز چنانچه شرط دارای ارزش درستی باشد دستورات ۱ تا n اجرا میشوند و گرنه دستورات e1 تا en.

مثال۱ :

```
if (x >= y)
    max = x;
else
    max = y;
```

مثال۲ : هرگاه بخواهیم دو یا چند دستور را درون if یا else قرار دهیم از آکولاد باز و بسته { } استفاده می‌کنیم.

```
if (x > y)
{
    x++;
    max = x;
}
else
{
    y++;
    max = y;
}
```

دستور if تو در تو :

اگر بخواهیم از دستور if برای تست شرطهای متعددی استفاده کنیم باید آنها را به طور تودرتو بنویسیم. ساختار if تو در تو به صورت زیر است :

```
if ( شرط ۱ )
        دستور ۱;
else if ( شرط ۲ )
        دستور ۲;
.
.
else if ( شرط n )
        دستور n;
else
        دستور else;
```

در ساختار بالا اگر شرط۱ درست باشد دستور ۱ اجرا میشود در غیر اینصورت اگر شرط ۲ درست باشد دستور ۲ اجرا میشود و ...
در ساختار بالا میتوان آخرین else را حذف کرد.

۳- عملگرهای رابطه ای و منطقی در دستورات شرطی :

در دستورات شرطی میتوانیم از عملگرهای رابطه ای برای ارتباط بین متغیرها و متغیرهای کنترل و از عملگرهای منطقی برای ترکیب کردن شرط استفاده کرد. برای مثال اگر بخواهیم شرطی بنویسیم که در آن x از y بزرگتر و y از z بزرگتر باشد باید به صورت زیر عمل کنیم :

```
if ( (x>y) && (y>z) )
```

در مثال فوق از عملگر منطقی && که همان 'and' است استفاده کردیم. اگر می‌خواستیم شرطی بنویسیم که درآن x از y بزرگتر یا y از z بزرگتر باشد باید به جای && از علامت || که همان 'or' است استفاده می‌کردیم.

مثال : میخواهیم برنامه ای بنویسیم که کاربر عددی را وارد کند و با زدن دکمه Apply برنامه به ما بگوید که عدد مثبت ، منفی و یا صفر است.


```
private void button1_Click(object sender, EventArgs e)
{
    int num = Convert.ToInt۳۲(textBox1.Text);
    if (num > 0)
        MessageBox.Show("عدد مثبت است");
    else if (num < 0)
        MessageBox.Show("عدد منفی است");
    else
        MessageBox.Show("عدد صفر است");
}
```

برنامه را اجرا کنید و یک عدد جلوی textBox1 بنویسید و دکمه Apply را بزنید تا برنامه پاسخ مناسب را بدهد.

4– **معرفی کنترلهای** CheckBox **و** RadioButton : هرگاه بخواهیم گزینه‌هایی را در اختیار کاربر قرار دهیم تا کاربر آنها را بر اساس اختیار خود انتخاب کند از این دو کنترل استفاده می‌کنیم.

کنترل دکمه رادیویی(RadioButton): هرگاه بخواهیم چند گزینه را در اختیار کاربر قرار دهیم به صورتی که کاربر فقط بتواند یکی از گزینه‌ها را انتخاب کند از این کنترل استفاده می‌کنیم. برخی از خواص ، رویدادهای مهم این کنترل در سه جدول زیر آمده است :

برخی از خواص مربوط به دکمه رادیویی (RadioButton)

خاصیت	توضیحات
Name	نام کنترل را برای کدنویسی مشخص می‌کند
Checked	اگر این خاصیت True شود دکمه رادیویی انتخاب می‌شو

برخی از رویدادهای مربوط به دکمه رادیویی (RadioButton)

خاصیت	توضیحات
Checked_Changed	این رویداد زمانی رخ می‌دهد که حالت انتخاب دکمه رادیویی تغییر کند (چه انتخاب شود و چه از حالت انتخاب درآید)

کنترل جعبه انتخاب (CheckBox) : هرگاه بخواهیم کاربر یک یا چند گزینه را انتخاب نماید از این کنترل استفاده می‌کنیم. هر کنترل CheckBox برخلاف RadioButton ، به صورت مستقل عمل می‌نماید و انتخاب و عدم انتخاب یک CheckBox بر روی CheckBox های دیگر تأثیری ندارد. برخی از خواص ، رویدادهای مهم این کنترل در سه جدول زیر آمده است :

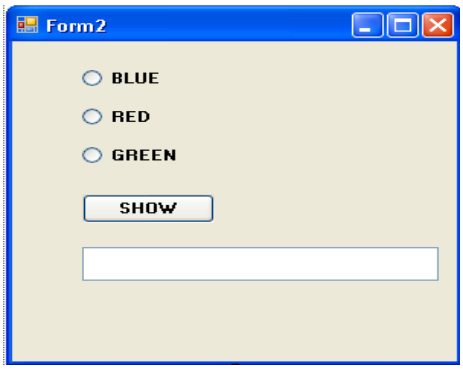
برخی از خواص مربوط به جعبه انتخاب (CheckBox)

خاصیت	توضیحات
Name	نام کنترل را برای کدنویسی مشخص می‌کن
Checked	اگر این خاصیت True شود جعبه انتخاب می‌شود
CheckState	حالت جعبه انتخاب را مشخص می‌کند. این خاصیت سه مقدار UnChecked ، Checked و Indeterminate را می‌پذیرد

برخی از رویدادهای مربوط به جعبه انتخاب (CheckBox)

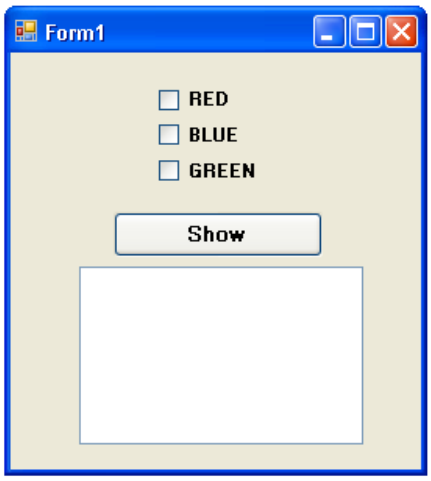
خاصیت	توضیحات
Checked_Changed	این رویداد زمانی رخ می‌دهد که حالت انتخاب جعبه انتخاب تغییر کند (چه انتخاب شود و چه از حالت انتخاب درآید)

مثال: برنامه ای نویسید که دارای ۳ دکمه رادیویی به شکل زیر باشد که کاربر بتواند در هر لحظه یکی از این دکمه‌ها را انتخاب کند.
اگر کاربر روی دکمه Show کلیک کند متن دکمه‌ای که انتخاب شده درون جعبه متن نمایش داده شود.




```
private void btnShow_Click(object sender, EventArgs e)
{
   if (radioButton1.Checked == true)
      textBox1.Text = radioButton1.Text;
   else if (radioButton2.Checked == true)
      textBox1.Text = radioButton2.Text;
   else if (radioButton3.Checked == true)
      textBox1.Text = radioButton3.Text;
   else
      textBox1.Text = "All radio buttion is not checked";
}
```

مثال: برنامه ای نویسید که دارای ۳ جعبه رادیویی به شکل زیر باشد که کاربر بتواند هرکدام را که می‌خواهد انتخاب کند.
اگر کاربر روی دکمه Show کلیک کند متن دکمه‌هایی که انتخاب شده‌اند، هر کدام در یک سطر از جعبه متن نمایش داده شوند. خاصیت Multiline و AcceptReturn مربوط به جعبه متن را با true تنظیم کنید.




```
private void btnShow_Click(object sender, EventArgs e)
{
   textBox1.Clear();
   if (checkBox1.Checked == true)
      textBox1.AppendText("RED\n");
   if (checkBox2.Checked == true)
      textBox1.AppendText("BLUE\n");
   if (checkBox3.Checked == true)
      textBox1.AppendText("GREEN\n");
}
```

سؤال: چرا در مثال مربوط به دکمه رادیویی از if else استفاده کردیم ولی در مثال دومی فقط از ifهایی استفاده کردیم که به هم ربطی ندارند؟ چون انتخاب یا عدم انتخاب دکمه‌های رادیویی به هم مربوطند و بایستی شرط انتخاب آنها در دستورات تصمیم‌گیری نیز به هم مربوط باشند.

۵- ساختار تصمیم switch :

ساختار switch یکی از ساختارهای جالب و مهم است. از این ساختار برای تصمیم گیریهای چندگانه بر اساس مقادیر مختلف یک عبارت استفاده میشود . به طور کلی در تمام تصمیم گیریهایی که بیش از سه انتخاب وجود دارد بهتر است از ساختار switch استفاده شود. این ساختار به صورت زیر بکار میرود :

```
switch ( عبارت )
{
      case < مقدار۱ > :
                  < دستورات۱ >
                  break;
      case < مقدار۲ > :
                  < دستورات۲ >
                  break;
      ⋮
      case < nمقدار > :
                  < n-1 دستورات >
                  break;
      default :
                  < n دستورات >
}
```

عملکرد این ساختار بدین صورت است که که مقدار عبارت جلوی switch با مقدار جلوی هر case بررسی میشود اگر با هر کدام برابر باشد دستورات مربوط به آن case اجرا میشود.

تذکر : برای اینکه بعد از اجرای دستورات جلوی هر case از ساختار switch خارج شویم از دستور break استفاده میکنیم.

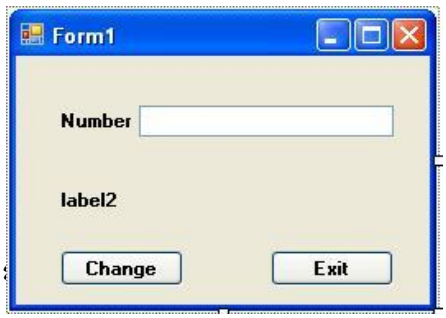
نکاتی در مورد ساختار switch :

۱- ساختار switch میتواند فاقد بخش default باشد. در اینصورت اگر عبارت محاسبه شده با هیچکدام از مقادیر ذکر شده برابر نباشد هیچکدام از دستورات اجرا نمیشوند.

۲- مقادیر موجود در case های switch نمیتوانند با هم مساوی باشند.

۳- اگر در یک case از break استفاده نشود با case بعدی or می‌شود.

مثال : برنامه ای بنویسید که رقمی را از کاربر بین 0 تا 9 بگیرد و معادل آن را به لاتین چاپ کند (مانند 7 که می‌شود Seven)



۲

```
private void btnChange_Click(object sender, EventArgs e)
{
    switch (Convert.ToInt32(textBox1.Text))
    {
        case 0: label2.Text = "Ziro";  break;
        case 1: label2.Text = "One";   break;
        case 2: label2.Text = "Two";   break;
        case 3: label2.Text = "Three"; break;
        case 4: label2.Text = "Four";  break;
        case 5: label2.Text = "Five";  break;
        case 6: label2.Text = "Six";   break;
        case 7: label2.Text = "Seven"; break;
        case 8: label2.Text = "Eight"; break;
        case 9: label2.Text = "Nine";  break;
        default:label2.Text = "Number not in 0 .. 9"; break;
    }
}

private void btnExit_Click(object sender, EventArgs e)
{
    Close();
}
```

۶- **معرفی کنترلهای جعبه لیست** (ListBox) **و جعبه کشویی** (ComboBox)**:** از این دو کنترل می‌توان برای اینکه بخواهیم مقادیری را درون لیست نگهداری کنیم تا کاربر بتواند یکی یا چند مقدار را از لیست انتخاب کند استفاده می‌شود.

کنترل جعبه لیست (ListBox) : با استفاده از این لیست میتوان لیستی را ایجاد کرد و اطلاعاتی راجع به لیست کتابها ، لیست افراد ، لیست شهرها و غیره را در آن قرار داد. عناصر موجود در listBox میتوانند به طرف بالا و پائین حرکت کنند. بعضی از خواص و رویدادها و متدهای این کنترل عبارتند از :

برخی از خواص مربوط به جعبه لیست (ListBox)

خاصیت	توضیحات
Name	نام کنترل را برای کدنویسی مشخص می‌کن
Items	با این خاصیت میتوان گزینه‌هایی را تعیین کرد تا در کنترل ظاهر شوند
MultiColumn	تعیین میکند آیا کنترل میتواند چند ستون داشته باشد.
SlectionMode	تعیین میکند که آیا در کنترل listBox میتوان چند گزینه را انتخاب کرد.
Sorted	تعیین میکند که آیا گزینه ها کنترل listBox به صورت مرتب شده باشند یا خیر.
SelectedIndex	ای را برمیگرداند که باید انتخاب شود. شماره گزینه های listBox از صفر شروع میشود
SelectedItems	برای حالتی به کار میرود که چند گزینه را بتوان انتخاب کرد
SelectedValue	مقداری را تعیین میکند که گزینه مربوط به آن باید انتخاب شود
Count	این گزینه تعداد گزینه‌های مربوط به جعبه لیست را مشخص می‌کند. مثال : `x = listBox1.Items.Count;`

برخی از رویدادهای مهم مربوط به جعبه لیست (ListBox)

خاصیت	توضیحات
SelectedValueChanged	این رویداد زمانی رخ می‌دهد که کاربر مقدار انتخاب شده جعبه لیست را تغییر دهد (یا مقداری دیگر را انتخاب کند)

برخی از متدهای مهم مربوط به جعبه لیست (ListBox)

خاصیت	توضیحات
Clear( )	این متد تمامی گزینه‌های مربوط به جعبه لیست را پاک می‌کند. `listBox1.Items.Clear()`
Add( )	برای اضافه کردن گزینه ای به انتهای لیست بکار میرود ; `listBox1.Items.Add ( "Ziro" )`
Insert(index, item)	برای اضافه کردن گزینه جدید در مکان مشخصی بکار می‌رود ; `listBox1.Items.Insert(2,"mehdi")`
Remove (متن)	برای حذف متن مشخصی که درون لیست وجود دارد بکار می‌رود ; `listBox1.Items.Remove("mehdi")`
RemoveAt(index)	برای حذف گزینه ای توسط شماره اندیس جعبه لیست بکار می‌رود. ; `listBox1.Items.RemoveAt( 3 )`

کنترل جعبه کشویی (ComboBox) : با استفاده از این جعبه کشویی میتوان لیستی را ایجاد کرد و اطلاعاتی راجع به لیست کتابها ، لیست افراد ، لیست شهرها و غیره را در آن قرار داد ولی از این لیست فقط میتوان یک گزینه را انتخاب کرد و نشان داد. تمامی خصوصیات و رویدادها و متدهایی که برای جعبه لیست گفته شد در این کنترل نیز صادق است.

مثال : برنامه ای بنویسید که فرم آن به شکل زیر باشد. و کاربر بتواند با زدن دکمه Add گزینه‌ای که در textbox1 نوشته می‌شود را به جعبه‌لیست اضافه و با زدن دکمه Remove گزینه‌ای که کاربر از لیست انتخاب کرده را حذف و با زدن دکمه Remove by Index گزینه‌ای را که شماره آن را درون textBox2 می‌نویسیم را حذف کند.


```
private void btnAdd_Click(object sender, EventArgs e)
{
    if (textBox1.Text != "")
       listBox1.Items.Add(textBox1.Text);
}
private void btnRemove_Click(object sender, EventArgs e)
{
    if (listBox1.SelectedIndex != -1)
       listBox1.Items.RemoveAt(listBox1.SelectedIndex);
    else
       MessageBox.Show("لطفا گزینه ای را انتخاب کنید");
}
private void btnRemoveByIndex_Click(object sender, EventArgs e)
{
    int tedad = listBox1.Items.Count;
    int x_del = Convert.ToInt32(textBox2.Text);
    if (x_del >= tedad || x_del < 0)
       MessageBox.Show("عددی که برای حذف وارد کرده اید خارج از محدوده است");
    else
       listBox1.Items.RemoveAt(x_del);
}
```

در خط هشتم : اگر درون `listBox` **گزینه‌ای انتخاب نشود خاصیت** `SelectedIndex` **آن برابر** `-1` **می‌شود.**

۷- **معرفی جعبه پیغام (MessageBox) :** از این جعبه برای نمایش پیغام ، پرسیدن سئوال و ... استفاده می‌شود. نحوه استفاده از این تابع به صورت زیر است :

```
MessageBox.Show("Text","caption",MessageBoxButtons, MessageBoxIcon);
```

`Text`: متنی که میخواهیم درون جعبه متن ظاهر شود را مشخص می‌کند.

`caption`: عنوان جعبه متن را مشخص می‌کند.

`MessageBoxButtons`: دکمه‌هایی که بایستی درون جعبه متن ظاهر شوند را مشخص می‌کند. مثل ← `MessageBoxButtons.OK`

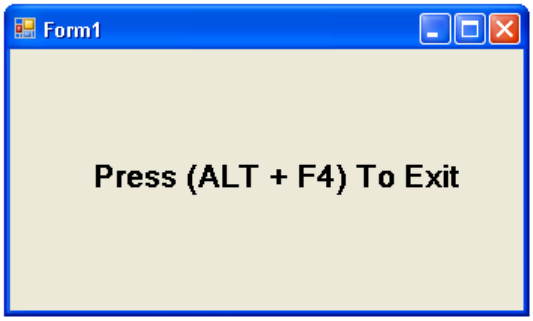
`MessageBoxIcon`: آیکونی که باید درون جعبه متن ظاهر شود را مشخص می‌کند. مثل ← `MessageBoxIcon.Information`

جعبه پیغامی که نمایش داده می‌شود ، دکمه‌هایی دارد که ما در کدنویسی آنها را مشخص می‌کنیم و بایستی مدیریت شوند ، چون این دکمه‌ها به تنهایی هیچ کاری را انجام نمی‌دهند. در مثال بعدی طرز کار اینگونه جعبه پیغام‌ها آورده شده است.

مثال کاربردی : برنامه ای بنویسید که درون یک فرم اگر Alt+F4 فشرده شد ، جعبه پیغامی ظاهر شود که از کاربر سئوال بپرسد "آیا میخواهد از برنامه خارج شوید؟" اگر کاربر دکمه Yes را کلیک کرد از برنامه خارج شود ، در غیر اینصورت خارج نشود.

حل : بایستی جعبه پیغام دارای دو دکمه Yes و No باشد. برای اینکه بفهمیم که کاربر کدام دکمه را فشار داده از کلاس DialogResult استفاده می‌کنیم ، این کلاس تمامی دکمه‌هایی که توسط کاربر فشرده می‌شود را می‌تواند تشخیص دهد.

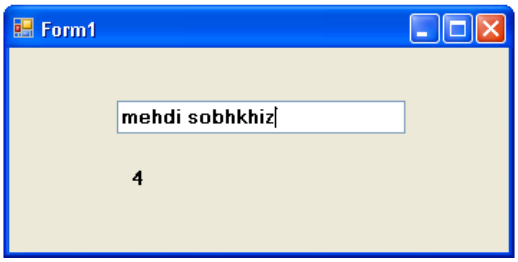
برای اینکه کلید Alt+F4 را تشخیص دهیم از رویداد KeyDown مربوط به فرم استفاده می‌کنیم. اگر کاربر این دکمه‌ها را فشار داد بایستی فشردن دکمه را لغو نمود که این کار با true کردن Handled مربوط به دکمه انجام می‌شود (e.Handled=true). خاصیت KeyPreview مربوط به فرم را true کنید ، چون اگر فرم دارای کنترلهای دیگری باشد ، تست فشردن دکمه‌ها از دست فرم خارج می‌شود.




```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
   if (e.Alt == true && e.KeyCode == Keys.F4)
   {
      DialogResult dr = new DialogResult();
      dr = MessageBox.Show("آیا میخواهید از برنامه خارج شوید؟", "خروج",
                           MessageBoxButtons.YesNo, MessageBoxIcon.Question);
      if (dr == DialogResult.Yes)
         Close();
      else
         e.Handled = true;
   }
}
```

مثال : برنامه ای بنویسید که دارای یک textbox باشد که اگر کاربر درون این جعبه متن کلیدهای صدادار انگلیسی را فشرد یک واحد به مقدار Sum اضافه شود. (Sum یک متغیر عمومی در برنامه با مقدار صفر اولیه تعریف می‌کنیم).

میخواهیم در این برنامه تعداد کلیدهای صدادار فشرده شده را توسط کاربر در Sum ذخیره کنیم ، هر زمان که کلید صدادار جدید فشرده می‌شود ، مقدار جدید Sum درون یک برچسب نمایش داده شود.



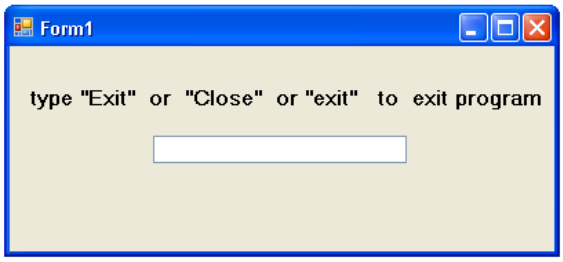

```
int Sum = 0;     // متغیر عمومی
private void textBox1_KeyDown(object sender, KeyEventArgs e)
{
    if ( e.KeyCode == Keys.A || e.KeyCode == Keys.E ||
         e.KeyCode == Keys.U || e.KeyCode == Keys.O ||
         e.KeyCode == Keys.I || )
    {
        Sum++;
        label1.Text = Sum.ToString();
    }
}
```

برای اینکه بتوانیم بفهمیم که کاربر دکمه صدادار را فشرده است از **Or** کردن دکمه‌های صدادار استفاده کردیم.

متغیر عمومی بایستی قبل از تابع KeyDown تعریف شود. نیازی به صفر کردن آن نیست چون متغیر عمومی خودش با مقدار اولیه صفر تنظیم می‌شود.

مثال : برنامه ای بنویسید که دارای فقط یک textbox باشد که اگر کاربر درون این جعبه متن ، کلماتی مثل "Exit" یا "Quit" یا "Close" نوشت از برنامه خارج شود.

حل: بایستی درون رویداد TextChanged کدمان را بنویسیم ، یعنی هر زمان که متن تغییر داده شد رویداد فراخوانی شود.




```
private void textBox1_TextChanged(object sender, EventArgs e)
{
     string text = textBox1.Text.ToLower();
     if (text == "exit" || text == "quit" || text == "close")
        Close();
}
```

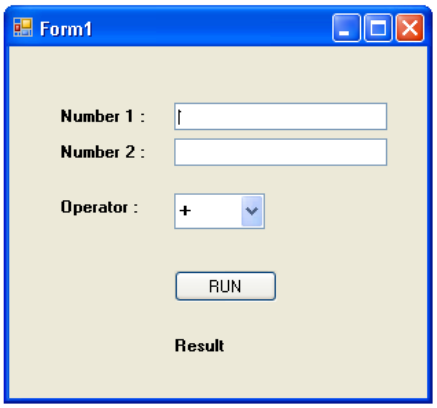
تابع ToLower() برای این است که متن وارد شده را به حروف کوچک تبدیل کند. چون ممکن است کاربر حروف را بزرگ یا کوچک وارد کند و بین حروف کوچک و بزرگ تفاوت است. از تابع ToUpper() نیز می‌توانستیم برای تبدیل رشته به حروف بزرگ استفاده کنیم که در این صورت در شرط بایستی کلمات خروج را با حروف بزرگ می‌نوشتیم.

مثال : برنامه ای بنویسید که دارای دو textBox برای ورود اعداد صحیح یا اعشاری و یک comboBox و یک برچسب با نام lblResult برای نمایش نتیجه و یک دکمه برای اجرا باشد. گزینه‌های comboBox را با "+" ، "-" ، "/" و "*" پر کنید. بر اساس گزینه‌ای که کاربر از comboBox انتخاب می‌کند ، کد لازم را برای انجام محاسبات و نمایش بنویسید.

این برنامه چهار عمل اصلی را پیاده سازی می‌کند. برای قرار دادن گزینه‌های + - / * درون comboBox از جعبه خواص ، خاصیت Items را کلیک کنید و از دکمه سه نقطه جلوی Collection استفاده کنید. کادری باز می‌شود که می‌توانید هر کدام از گزینه‌ها را درون comboBox قرار دهید. خاصیت Text را نیز با + پر کنید.

اگر میخواهید از طریق کدنویسی گزینه‌ها را پر کنید بایستی در رویداد Load مربوط به فرم به صورت زیر بنویسید :

```
private void Form1_Load(object sender, EventArgs e)
{
     comboBox1.Items.Clear();
     comboBox1.Items.Add("+");
     comboBox1.Items.Add("-");
     comboBox1.Items.Add("*");
     comboBox1.Items.Add("/");
     comboBox1.Text = "+";
}
```




```
private void btnRun_Click(object sender, EventArgs e)
{
  Double x = Convert.ToDouble(textBox1.Text);
  Double y = Convert.ToDouble(textBox2.Text);
  switch (comboBox1.Text)
  {
    case "+": lblResult.Text = (x + y).ToString(); break;
    case "-": lblResult.Text = (x - y).ToString(); break;
    case "*": lblResult.Text = (x * y).ToString(); break;
    case "/": lblResult.Text = (x / y).ToString(); break;
    default:  lblResult.Text = "ERROR"; break;
  }
}
```

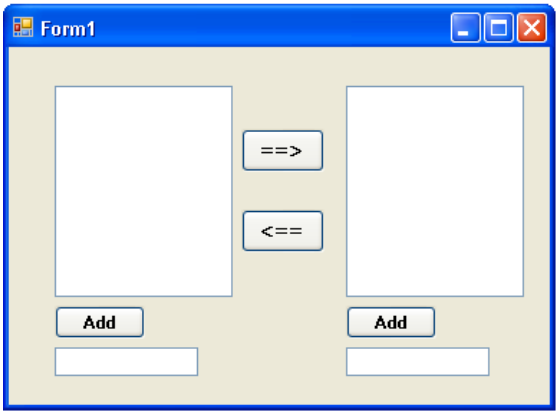
۸- تمرین در منزل

۱- برنامه‌ای بنویسید که دارای یک **comboBox** با گزینه‌های **"Red"** و **"Blue"** و **"Green"** باشد که اگر کاربر یکی از این گزینه‌ها را انتخاب کرد رنگ پس زمینه فرم به آن رنگ در آید . (راهنمایی از رویداد SelectedIndexChanged مربوط به **comboBox** استفاده کنید).

مثلا برای تغییر رنگ فرم به قرمز داریم

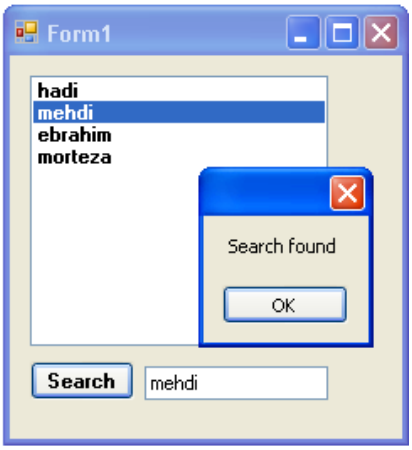
```
Form1.ActiveForm.BackColor = Color.Red;
```

۲- برنامه‌ای بنویسید که دارای دو listBox باشد که هر کدام دارای دکمه‌های اضافه کردن متن به listBoxهای خودشان باشند (یک textBox برای هر کدام نیز بگذارید). هر کدام دارای دکمه‌ای باشند که اگر کاربر درون listBox گزینه‌ای را انتخاب کرد دکمه‌اش زده شد ، گزینه از listBox حذف شود و به listBox دیگری انتقال یابد. سعی کنید برنامه‌تان خطایی نداشته باشد. و خطاهایی که ممکن است اتفاق بیفتند را درون جعبه پیغام نمایش دهید. فرم را به شکل زیر طراحی کنید .



۳- تمرین حل شده :

برنامه ای که دارای یک **listBox** ، یک جعبه متن و یک دکمه باشد. که اگر کاربر درون **textBox** متنی وارد کرد ، اگر درون **listBox** وجود داشت یک پیغام مبنی بر اینکه یافت شد بدهد در غیر اینصورت پیغام یافت نشد دهد.(توسط جعبه پیغام ، پیغامها نمایش داده شوند). اگر متن یافت شد درون **listBox** انتخاب شود.




```
private void button1_Click(object sender, EventArgs e)
{
   int index = listBox1.FindString(textBox1.Text);
   if (index != -1)
   {
      listBox1.SelectedIndex = index;
      MessageBox.Show("Search found");
   }
      else
   {
      listBox1.SelectedIndex = -1;
      MessageBox.Show("Search not found");
   }
}
```

فصل چهارم : ساختارهای تکرار

۱- ساختارهای تکرار
۲- ساختار تکرار for
۳- ساختار تکرار while
۴- ساختار تکرار do
۵- دستور break و continue
۶- ساختار تکرار foreach
۷- کنترل MsFlexGrid
۸- کنترل DataGridView
۹- کار با رشته‌ها
۱۰-نوع داده تاریخ و زمان
۱۱-تمرین در خانه

ساختارهای تکرار :

ماهیت بسیاری از کارها تکراری است. مثل خواندن اسامی تعدادی دانشجو ، خواندن ۱۰ عدد صحیح و محاسبه مجموع آنها.
در C#.Net برای انجام کارهایی که ماهیت آنها تکراری است از حلقه های تکرار استفاده میشود. حلقه های تکرار عبارتند از: **for** ، **while** ،**do**
. **foreach** ،

۱– ساختار تکرار for :

ساختار تکرار for یکی از امکانات ایجاد حلقه است و معمولاً در حالتی به کار میرود که تعداد دفعات تکرار حلقه از قبل مشخص باشد. در این ساختار متغیری وجود دارد که تعداد دفعات تکرار را کنترل میکند. این متغیر شمارنده یا اندیس حلقه نام دارد. اندیس حلقه دارای مقدار اولیه یا گام اولیه میباشد که در هر بار اجرای دستورات حلقه مقداری به این اندیس اضافه میشود. یکی دیگر از اجزای حلقه شرط حلقه میباشد. شرط حلقه مشخص میکند که تا کی باید حلقه ادامه پیدا کند. اگر شرط درست باشد دستورات حلقه اجرا میشوند و گرنه کنترل برنامه از حلقه خارج میشود.

دستور for را به دو شکل زیر میتوان نوشت :

شکل اول

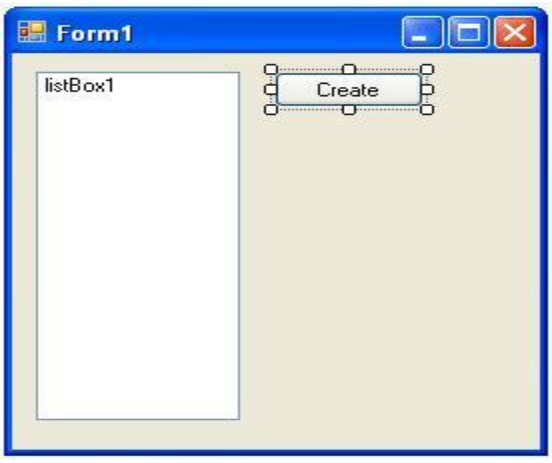
```
for ( مقدار اولیه اندیس حلقه ;  شرط حلقه  ;  گام حرکت  )
{
        دستور ۱ ;
        دستور۲ ;
        :
        دستور n ;
}
```

شکل دوم

```
for ( ; ; )
{
        دستور ۱ ;
        دستور۲ ;
        :
        دستور n ;
}
```

نکته : در شکل دوم چون هیچ شرطی برای خاتمه دادن به حلقه وجود ندارد. حلقه تکرار تا بینهایت ادامه می یابد.

مثال۱ : میخواهیم برنامه ای بنویسیم که اعداد یک تا بیست را در کنترل listBox ذخیره کند.




```
private void button1_Click(object sender, EventArgs e)
{
    for (int i = 1; i <= 20; i++)
       listBox1.Items.Add( i.ToString() );
}
```

نکاتی در مورد حلقه تکرار for :

- اندیس حلقه می‌تواند صحیح ، اعشاری و کارکتری باشد.
- اگر حلقه تکرار for به سیمی‌کولن (;) ختم شود ، دستور بعد از حلقه for فقط یکبار اجرا می‌شود.

۲– ساختار تکرار while :

ساختار تکرار while یکی دیگر از امکاناتی است که برای اجرای دستورات بکار می‌رود. این ساختار به صورت‌های زیر بکار می‌رود :

1. while (**شرط**)
   دستور;

2. while (**شرط**)
   {
   دستور ۱;
   دستور 2;
   :
   دستور n;
   }

همانطوریکه ملاحظه می‌کنید وقتی دستورات تکرار شونده بیش از یکی باشند باید آنها را در بین آکولاد باز و بسته قرار دهیم. در حلقه while اگر شرط حلقه درست باشد ، دستورات حلقه اجرا می‌شوند و گرنه کنترل برنامه از حلقه خارج می‌شود.

برای اینکه حلقه خاتمه پیدا کند ، شرط باید در داخل حلقه نقض شود. اگر شرط حلقه همیشه درست باشد و هیچگاه نقض نشود حلقه تکرار بینهایت بار اجرا می‌شود.

۳– ساختار تکرار do :

ساختار تکرار do مانند ساختار تکرار while است. با این تفاوت که در ساختار while شرط حلقه در ابتدای حلقه تست می‌شود ، در حالیکه در ساختار do شرط حلقه در انتهای حلقه تست می‌گردد. بنابراین ، دستورات موجود در حلقه do حداقل یکبار اجرا می‌شوند.. این ساختار به صورت‌های زیر بکار می‌رود :

1. do
   دستور;
   while (**شرط**);

2. do
   {
   دستور ۱;
   دستور 2;
   :
   دستور n;
   } while (**شرط**);

۴– دستور break :

این دستور موجب خروج از حلقه‌های تکرار می‌شود و نحوه کاربرد آن به صورت زیر است :

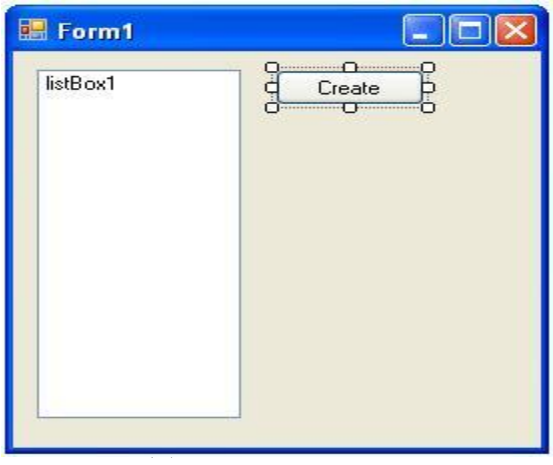
break ;

اگر چند حلقه تودرتو وجود داشته باشد ، این دستور موجب خروج از داخلی ترین حلقه تکرار می‌شود. کاربرد دیگر این دستور ، خاتمه دادن به ساختار switch بود.

۵– دستور continue :

این دستور در حلقه تکرار موجب انتقال کنترل برنامه به ابتدای حلقه می‌شود. پس از انتقال کنترل به ابتدای حلقه شرط حلقه مورد بررسی قرار می‌گیرد ، چنانچه شرط درست باشد ، اجرای دستورات حلقه ادامه می‌یابد و گرنه حلقه تکرار خاتمه می‌یابد.

مثال۲ : می‌خواهیم برنامه ای بنویسیم که اعداد ۱ تا ۱۰۰ را در کنترل listBox ذخیره کند (اعدادی که بر ۷ بخش پذیر هستند را نمایش ندهد).




```
private void button1_Click(object sender, EventArgs e)
{
   for (int i = 1; i <= 100; i++)
   {
       if (i % 7 == 0)
          continue;
       else
          listBox1.Items.Add(i.ToString());
   }
}
```

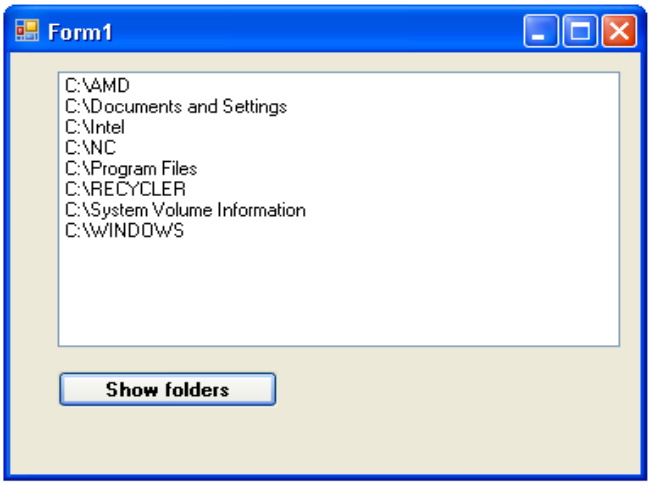
تمرین : برنامه مثال قبلی را با حلقه‌های تکرار while و do نیز بنویسید.

۶- دستور foreach :

این دستور یکی از ساختارهای مهم برای تکرار دستورات می‌باشد و معمولا برای تکرار مقادیر همنوع و همگن از قبیل آرایه‌ها و ... بکار می‌رود. از این دستور کمتر از دستور for استفاده می‌شود ولی در بعضی مواقع کار ما را بسیار پیش می‌اندازد. طرز استفاده از این دستور به صورت زیر است :

```
foreach ( متغیر   in   مجموعه مقادیر )
{
    دستورات;
}
```

مثال۳ : برنامه‌ای بنویسید که کلیه فولدرهای درایو C را درون listBox1 نمایش دهد.




```
private void button1_Click(object sender, EventArgs e)
{
  foreach (String Folder in System.IO.Directory.GetDirectories("C:\\"))
       listBox1.Items.Add(Folder.ToString());
}
```

نکته : در صورتیکه تعداد دفعات تکرار مشخص باشد از حلقه for استفاده کنید در غیر این صورت از while یا do استفاده کنید. اگر میخواهید دستورات حداقل یکبار اجرا شوند از do استفاده کنید.

۷- کنترل MsFlexGrid :

با این کنترل می‌توان اطلاعات را به صورت جدول نشان داد. همانطوریکه می‌دانید جدول از تعدادی سطر و ستون تشکیل شده است. محل برخورد یک سطر و ستون ، خانه یا سلول نام دارد. بسیاری از خواص این کنترل در زمان اجرا مفید هستند. برخی از خواص ، رویدادها و متدهای این کنترل در زیر آمده است.

برخی از خواص مربوط به MsFlexGrid

خاصیت	توضیحات
Name	نام کنترل را برای کدنویسی مشخص می‌کند
BackColorBkg	رنگ پس زمینه کنترل را مشخص می‌کند
BackColoFixed	با این خاصیت می‌توان رنگ زمینه مربوط به سطر و ستون ثابت را تعیین کرد.
BackColorSel	رنگ پس زمینه مربوط به سلول انتخاب شده را تعیین می‌کند.
Cols	تعداد ستونهای کنترل را تعیین می‌کند
Rows	تعداد سطرهای کنترل را تعیین می‌کند
FixedCols	تعداد ستونهای ثابت را تعیین می‌کند
FixedRows	تعداد سطرهای ا تعیین می‌کند
ForeColorFixed	رنگ متن سطر و ستون ثابت را تعیین می‌کند
ForeColorSel	رنگ متن خانه انتخاب شده را تعیین می‌کند
GridColor	رنگ خطوط شبکه‌ای را تعیین می‌کند

GridColorFixed	رنگ خطوط شبکه‌ای سطر و ستون ثابت را تعیین می‌کند
GridLines	سبک نمایش خطوط شبکه‌ای را تعیین می‌کند
GridLinesFixed	سبک نمایش خطوط شبکه‌ای سطر و ستون ثابت را تعیین می‌کند
GridLineWidth	برای تعیین ضخامت خطوط شبکه‌ای بکار می‌رود
Col	برای تعیین موقعیت ستون فعلی بکار می‌رود
Row	برای تعیین موقعیت سطر فعلی بکار می‌رود

برخی از رویدادهای مربوط به MsFlexGrid

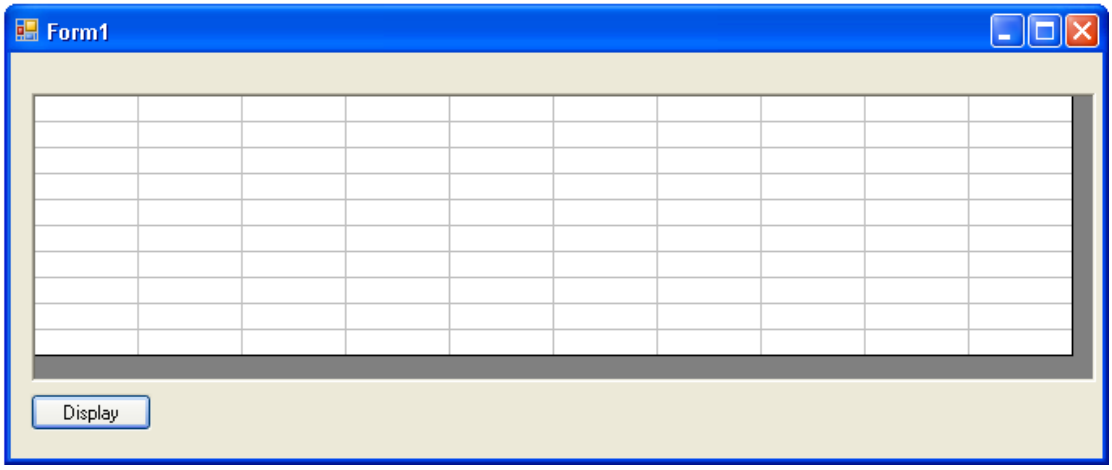
رویداد	توضیحات
EnterCell	اگر وارد خانه‌ای شوید این رویداد رخ می‌دهد
LeaveCell	اگر از خانه‌ای خارج شوید این رویداد رخ می‌دهد
RowColChanged	اگر خواص Row یا Col تغییر یابند این رویداد رخ می‌دهد

نکته : این کنترل بایستی به جعبه ابزار (ToolBox) اضافه شود. چون درون جعبه ابزار موجود نمی‌باشد. برای این‌کار روی جعبه ابزار راست کلیک کنید و از منوی باز شده گزینه Choose Items... را انتخاب کنید. از کادر باز شده ، سربرگ دوم یعنی Com Component را انتخاب و از لیست گزینه Microsoft Flex Grid Control 6 را انتخاب و دکمه OK را بزنید تا به جعبه ابزار اضافه شود.

مثال۴ : برنامه‌ای بنویسید که جدول ضرب 10×10 را درون کنترل MsFlexGrid نمایش دهد.

خاصیت های این کنترل را به صورت جدول زیر تنظیم کنید.

خاصیت	مقدار
Name	MFG1
Cols	10
Rows	10
FixedCols	0
FixedRows	0




```
private void button1_Click(object sender, EventArgs e)
{
   for (int i = 1; i <= 10; i++)
   {
      MFG1.Row = i - 1;
      for (int j = 1; j <= 10; j++)
      {
         MFG1.Col = j - 1;
         MFG1.Text = (i * j).ToString();
      }
    }
}
```

۸– کنترل DataGridView :

یکی از کنترلهای مهمی که برای نمایش جدول‌ها به خصوص جداول بانک اطلاعاتی بکار می‌رود کنترل DataGridView می‌باشد. برخی از خصوصیات و رویدادهای مهم این کنترل در جدولهای زیر آورده شده است (برخی دیگر در فصل بانکهای اطلاعاتی گفته خواهد شد).

برخی از خواص مربوط به DataGridView

خاصیت	توضیحات
Name	از این خاصیت برای کدنویسی استفاده می‌کنیم
AllowUserToAddRows	این خاصیت تعیین می‌کند که آیا کاربر می‌تواند سطر جدیدی به دیتاگرید اضافه کند یا خیر
AllowUserToDeleteRows	این خاصیت تعیین می‌کند که آیا کاربر می‌تواند سطری از دیتاگرید را حذف کند یا خیر
Columns	از این خاصیت می‌توان ستونهایی به دیتاگرید اضافه کرد و به هر ستون نام خاصی داد

برخی از رویدادهای مربوط به DataGridView

رویداد	توضیحات
CellBeginEdit	هنگامی که محتوای سلول خاصی را بخواهیم تغییر دهیم این رویداد رخ می‌دهد
CellClick	هنگامی که سلول خاصی کلیک شود این رویداد رخ می‌دهد
CellDoubleClick	هنگامی که سلول خاصی دبل کلیک شود این رویداد رخ می‌دهد
CellContentDoubleClick	هنگامی که محتوای سلول خاصی دبل کلیک شود این رویداد رخ می‌دهد
CellEndEdit	هنگامی که سلول خاصی تغییر یابد رخ می‌دهد
CellEnter	هنگامی که به یک سلول خاصی وارد شویم رخ می‌دهد
CellLeave	هنگامی که از سلول خاصی خارج شویم رخ می‌دهد
ColumnsAdded	هنگامی که به یک دیتاگرید ستونی اضافه شود رخ می‌دهد
ColumnsRemoved	هنگامی که از دیتاگرید ستونی حذف شود رخ می‌دهد
RowsAdded	هنگامی که به یک دیتاگرید سطری اضافه شود رخ می‌دهد
UserDeletingRow	هنگامی که از دیتاگرید سطری میخواهد حذف شود رخ می‌دهد
UserDeletedRow	هنگامی که از دیتاگرید سطری حذف شود رخ می‌دهد

مثال۵ : میخواهیم یک برنامه کوچک نسبتاً کامل که از دیتاگرید به صورت دستی استفاده می‌کند که کاربرد زیادی هم در پروژه‌های برنامه‌نویسی دارد بنویسیم.

در این مثال میخواهیم یک دیتاگرید طراحی کنیم که دارای ۴ ستون با سرستونهای (ColumnsHeader) زیر باشد :

نام سرستون یا Name	عنوان سرستون یا Header Text
ROWS	ردیف
NAME	نام
FAMILY	نام خانوادگی
HOURS	تعداد ساعت کار
HOURS_SALARY	حقوق ساعتی
SALARY	حقوق

برای ایجاد این سرستونها مراحل زیر را دنبال کنید :

ابتدا یک دیتاگرید به فرم اضافه کرده و روی سه نقطه مربوط به خاصیت Columns کلیک کنید تا کادر Edit Columns باز شود. از این کادر می‌توان ستونهای دیتاگرید را ویرایش کرد. روی دکمه Add کلیک کنید تا کادر Add Column باز شود. حال می‌توانید سرستونهای جدول بالا را به دیتاگرید اضافه کنید. برای مثال برای ایجاد سرستون "ردیف" با نام ''ROWS'' در قسمت Name ، ''ROWS'' و در قسمت Header ، ''ردیف'' را وارد کنید و دکمه Add بزنید تا این سرستون به دیتاگرید اضافه شود. بقیه سرستونها را نیز طبق جدول بالا به دیتاگرید اضافه کنید.

میخواهیم دیتاگرید را طوری طراحی کنیم که کاربر وقتی یک سطر به دیتاگرید اضافه می‌کند ، سرستون ردیف به صورت خودکار update شود و سلولهای مرتبط با HOURS و HOURS_SALARY و SALARY مقدار صفر بگیرند. برای اینکار بایستی در رویداد Rows_Added مربوط به دیتاگرید کد زیر را اضافه کنید :

```
private void dataGridView1_RowsAdded(object sender, DataGridViewRowsAddedEventArgs e)
{
    int i;
    for ( i = 1; i < dataGridView1.Rows.Count; i++)
       dataGridView1.Rows[i - 1].Cells["ROWS"].Value = i;
    dataGridView1.Rows[i - 1].Cells["HOURS"].Value = 0;
    dataGridView1.Rows[i - 1].Cells["HOURS_SALARY"].Value = 0;
    dataGridView1.Rows[i - 1].Cells["SALARY"].Value = 0;
}
```

در ابتدای اجرای برنامه خود دیتاگرید نیز یک سطر خالی دارد که بایستی سرستونهای آن update شوند ، برای اینکه به سرستونهای آن مقدار اولیه بدهیم از رویداد Load مربوط به فرم استفاده کنید و کد زیر را به برنامه اضافه کنید.

```
private void Form1_Load(object sender, EventArgs e)
{
    dataGridView1.Rows[0].Cells["ROWS"].Value = 1;
    dataGridView1.Rows[0].Cells["HOURS"].Value = 0;
    dataGridView1.Rows[0].Cells["HOURS_SALARY"].Value = 0;
    dataGridView1.Rows[0].Cells["SALARY"].Value = 0;
}
```

حال میخواهیم برنامه این قابلیت را داشته باشد که اگر کاربر سطری را از دیتاگرید میخواهد حذف کند از کاربر سئوال شود که آیا میخواهد این کار انجام شود یا خیر. اگر کاربر مایل به این کار بود سطر حذف شود و سرستون "ردیف" نیز دوباره update شود.

برای اینکار بایستی دو رویداد UserDeletingRow و UserDeletedRow مربوط به دیتاگرید را به صورت زیر به برنامه اضافه کنید.

```
private void dataGridView1_UserDeletingRow(object sender, DataGridViewRowCancelEventArgs e)
{
    DialogResult dr = new DialogResult();
    dr = MessageBox.Show("آیا میخواهید این رکورد را حذف کنید؟","حذف رکورد",
                         MessageBoxButtons.YesNo,MessageBoxIcon.Information);
    if (dr == DialogResult.No)
       e.Cancel = true;
}

private void dataGridView1_UserDeletedRow(object sender, DataGridViewRowEventArgs e)
{
    for (int i = 1; i < dataGridView1.Rows.Count; i++)
       dataGridView1.Rows[i - 1].Cells["ROWS"].Value = i;
}
```

حال میخواهیم که اگر کاربر مقادیری درون سلولهای HOURS ، HOURS_SALARY و یا SALARY مربوط به یک رکورد خاصی را دارد ویرایش می‌کند ، مقدار سلول SALARY مربوط به همان رکورد را update کند.

فرمول محاسبه حقوق به صورت زیر محاسبه می‌شود : SALARY = HOURS * HOURSE_SALARY

برای این محاسبه رویداد CellEndEdit را به صورت زیر به برنامه اضافه کنید :

```
private void dataGridView1_CellEndEdit(object sender, DataGridViewCellEventArgs e)
{
  if (e.ColumnIndex == 3 || e.ColumnIndex == 4 || e.ColumnIndex == 5)
  {
     Int32 X = Convert.ToInt32(dataGridView1.Rows[e.RowIndex].Cells["HOURS"].Value);
     Int32 Y = Convert.ToInt32(dataGridView1.Rows[e.RowIndex].Cells["HOURS_SALARY"].Value);
     dataGridView1.Rows[e.RowIndex].Cells["SALARY"].Value = (X * Y).ToString();
  }
}
```

این کنترل بسیار قدرتمند می‌باشد. این کنترل برای نمایش جدولهای بانک اطلاعاتی و گزارشگیری کاربرد فراوانی دارد و در بخش بانک اطلاعاتی دوباره کاربرد دیگر آن را خواهیم دید.

۹– کار با رشته‌ها :

برای اینکه با رشته ها بتوانیم کار کنیم بایستی با برخی از دستورات از پیش تعریف شده آشنا شویم.

دستورات زیر را همراه با مثال توضیح میدهیم :

یک رشته به نام str با مقدار روبرو تعریف می‌کنیم

```
string str = "Visual C#.Net";
```

- **دستور** Length : این دستور طول رشته را بر میگرداند.

```
int x = str.Length;
MessageBox.Show(x.ToString());
```

بعد از اجرای دستور بالا مقدار x برابر با ۱۳ میشود. چون طول رشته str برابر با ۱۳ است.

- **دستور** SubString : این دستور از یک رشته یک زیررشته را برمیگرداند.

```
string str2;
str2 = str.Substring(0, 6);
MessageBox.Show(str2);
```

خط اول یک متغیر رشته ای به نام str2 تعریف میکند. خط دوم با استفاده از دستور Substring از کارکتر صفرم به تعداد ۶ کارکتر از رشته str جدا میکند و در متغیر str2 ذخیره میکند. خط سوم متن درون str2 را در جعبه پیغام نمایش میدهد.

- **دستور** Format : این دستور رشته را قالب بندی میکند. و به صورت زیر بکار میرود :

```
String.Format("{0:x}", Var )
```

این دستور را با مثال توضیح میدهیم :

```
double dblNumber;
dblNumber = 13;
dblNumber /= 7;
MessageBox.Show(" بدون فرمت" + dblNumber);
MessageBox.Show("" + String.Format("{0:n5}", dblNumber) );
```

توضیح کد : خط اول یک متغیر اعشاری به نام dblNumber تعریف میکند. خط دوم مقدار ۱۳ را به این متغیر نسبت میدهد و در خط سوم این مقدار متغیر را بر ۷ تقسیم کرده و در خودش ذخیره می‌کند. خط چهارم بدون استفاده از دستور فرمت مقدار این متغیر را نمایش میدهد و در خط پنجم با استفاده از دستور فرمت مقدار این متغیر را نمایش میدهد.

دستور Format از دو بخش تشکیل می‌شود که یک بخش آن مربوط به آرگومامهای جایگزین متغیر است (که در بخش مربوط به Console گفته شده است). بخش دوم آن طرز نمایش متغیر می‌باشد. برای مثال n5 در مثال بالا بدین معناست که عدد اعشاری تا ۵ رقم اعشار گرد شود.

- **دستور** Replace : برای جایگزینی زیررشته ها بکار می رود.

```
string strData = "Hello Hadi sobhkhiz";
string strNewData;
strNewData = strData.Replace("Hadi", "Mehdi");
```

دستور سوم زیر رشته Hadi را در رشته strData پیدا کرده و با زیررشته Mehdi جایگزین میکند و در رشته strNewData ذخیره میکند.

مقدار strNewData بعد از دستور فوق برابر "Hello Mehdi sobhkhiz" میشود.

تبدیل نوع رشته‌ای به عدد صحیح یا اعشاری با استفاده از دستور Parse :

قبلاً با دستور Convert آشنا شدیم که این دستور میتوانست انواع مختلف داده ها را به یکدیگر تبدیل کند. حال با دستور جدیدی به نام Parse آشنا شویم که برای تبدیل رشته ای که شامل عدد باشد به عدد بکار میرود. طرز کار این دستور به صورت زیر است :

تبدیل رشته به عدد صحیح `int.Parse` (رشته یا متغیر رشته ای)

تبدیل رشته به عدد اعشاری `float.Parse` (رشته یا متغیر رشته ای)

تبدیل رشته به عدد اعشاری با دقت مضائف `double.Parse` (رشته یا متغیر رشته ای)

* البته Convert بسیار قویتر از Parse میباشد و میتواند تمامی انواع را به یکدیگر تبدیل کند.

۱۰– استفاده از تاریخ و زمان :

یکی دیگر از انواع داده ای که کاربرد زیادی دارد و احتمالاً از آن استفاده خواهید کرد تاریخ و زمان هستند. این نوع متغیرها یک تاریخ را در خود نگه می دارند. برای تعریف یک متغیر از نوع تاریخ به صورت زیر عمل میشود :

`DateTime` **نام متغیر;**

- مثال زیر تاریخ فعلی و زمان فعلی سیستم را نمایش میدهد :

```
DateTime dteDate;
dteDate = DateTime.Now;
MessageBox.Show(dteDate.ToString());
```

قالب بندی تاریخ و زمان :

در بالا یکی از قسمتهای قالب بندی تاریخ را دیدیم. اگر متغیری از نوع تاریخ را با استفاده از تابع ToString به رشته تبدیل کنیم ، تاریخ و زمان با هم نمایش داده خواهد شد. برای اینکه فقط تاریخ را نمایش دهد بایستی به جای() ToString از `ToShortDateString ()` استفاده می‌کردیم و برای اینکه فقط زمان را نشان می‌دادیم بایستی از `ToShortTimeString ()` استفاده می‌کردیم. در زیر برخی از قالبهای تاریخ و زمان همراه با توضیحات آورده شده است :

۱- Now : تاریخ فعلی و زمان فعلی سیستم را برمیگرداند.

۲- Today : تاریخ امروز و ساعت ۱۲ را برمیگرداند.

۳- Year : سال مربوط به تاریخ را برمیگرداند.

۴- Month : ماه مربوط به تاریخ را برمیگرداند.

۵- Day : روز مربوط به تاریخ را برمیگرداند

۶- Hour : ساعت را بر میگرداند.

۷- Minute : دقیقه را برمیگرداند.

۸- Second : ثانیه را برمیگرداند.

۹- DayOfWeek : نام روز هفته را به انگلیسی برمیگرداند.

۱۰- DayOfYear : چندمین روز سال را به عدد برمیگرداند.

۱۱- ToShortDateString : فقط تاریخ را نمایش میدهد.

۱۲- ToLongDateString : تاریخ و زمان را با هم نمایش میدهد.

کار با تاریخها و زمانهای خاص :

یکی از مواردی که کنترل آن همواره برای برنامه نویسان مشکل بوده است کار کردن با تاریخ و زمان بوده است. در این قسمت میخواهیم یک سری تابع برای دستکاری کردن تاریخ معرفی کنیم.

```
DateTime d1 = new DateTime(1387, 5, 14);
```

در بالا تاریخی را تعریف کردیم که مقدار آن را خودمان تنظیم کردیم.

در زیر چند تابع را معرفی میکنیم که با آنها میتوانید راحت با تاریخ کار کنید:

۱- AddDays : به تاریخ روز اضافه میکند. مثال : دستور روبر ۴ روز به تاریخ اضافه میکند `d1 = d1.AddDays(4)`

۲- AddMonths : به تاریخ ماه اضافه میکند. مثال : دستور روبر ۳ ماه به تاریخ اضافه میکند `d1 = d1.AddMonths(3)`

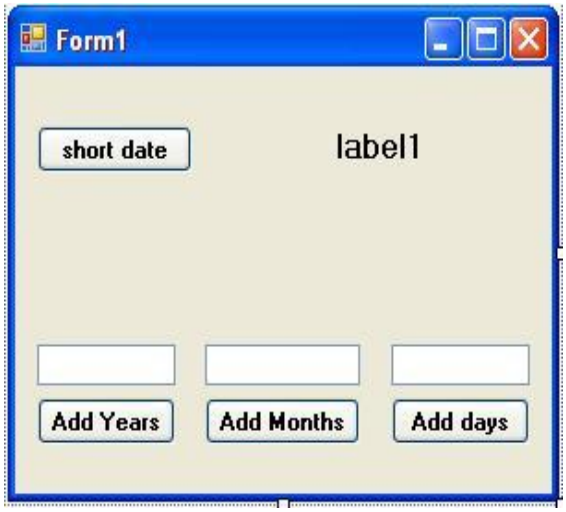
۳- AddYears : به تاریخ سال اضافه میکند مثال : دستور روبر ۷ سال از تاریخ کم میکند `d1 = d1.AddYears(-7)`

متدهای مهمی نیز برای کار با زمان وجود دارند که عبارتند از :

AddHours ، AddMinuts ، AddSeconds ، AddMiliseconds

مثال۶ : برنامه‌ای بنویسید که فرم آن به شکل زیر باشد. که با زدن دکمه short date تاریخ فعلی را در label1 نمایش دهد. با زدن دکمه Add Years مقداری که در textBox1 وجود دارد به آن مقدار به سال تاریخ فعلی اضافه کند و در label1 نمایش دهد و ...

نوع کنترل	خاصیت	مقدار
button	Name Text	btnShortDate short date
label	Name Text	lblDate label1
textBox	Name	txtYear
textBox	Name	txtMonth
textBox	Name	txtDay
button	Name Text	btnAddYears Add Years
Button	Name Text	btnAddMonth Add Months
button	Name Text	btnAddDays Add Days




```
private void btnShortDate_Click(object sender, EventArgs e)
{
    DateTime d1;
    d1 = DateTime.Now;
    lblDate.Text = d1.ToShortDateString();
}

private void btnAddYears_Click(object sender, EventArgs e)
{
    DateTime d1;
    d1 = DateTime.Now;
    d1 = d1.AddYears(Convert.ToInt32(txtYear.Text));
    lblDate.Text = d1.ToShortDateString();
}

private void btnAddMonth_Click(object sender, EventArgs e)
{
    DateTime d1;
    d1 = DateTime.Now;
    d1 = d1.AddMonths(Convert.ToInt32(txtMonth.Text));
    lblDate.Text = d1.ToShortDateString();
}

private void btnAddDays_Click(object sender, EventArgs e)
{
    DateTime d1;
    d1 = DateTime.Now;
    d1 = d1.AddDays(Convert.ToInt32(txtDay.Text));
    lblDate.Text = d1.ToShortDateString();
}
```

۱۱– تمرین در خانه :

۱– برنامه ای بنویسید که که از ورودی یک رشته را دریافت کند و از ورودی دیگر عددی صحیح مثل i را دریافت کند و با زدن یک دکمه کارکتر i ام رشته را حذف کند و رشته باقیمانده را چاپ کند.

۲– به فرم برنامه مثال ۵ یک برچسب و یک دکمه اضافه کنید که با کلیک بر روی دکمه مجموع حقوق کلیه اشخاص را درون برچسب نمایش دهد.

۳– برنامه‌ای بنویسید که تاریخ میلادی را از ورودی دریافت کند و آن را به هجری شمسی تبدیل کند و نمایش دهد.

فصل پنجم : آرایه ها

۱- آرایه ها
۲- تعریف و استفاده از آرایه‌ها
۳- روش مقدار دهی اولیه به آرایه
۴- آرایه‌های چندبعدی
۵- جستجو در آرایه
۶- مرتب سازی آرایه‌ها
۷- کلاس System.Array
۸- روش تولید کردن اعداد تصادفی و ذخیره آن در آرایه
۹- آرایه‌های دندانه‌دار (Jagged Array)
۱۰- شمارنده‌ها (enum)
۱۱-ساختار (struct)
۱۲-تمرین در خانه

۱– آرایه ها :

یکی از عمومی ترین نیازها در برنامه نویسی قابلیت نگهداری لیستی از اطلاعات مشابه و یا مرتبط به هم است. برای این منظور بایستی از آرایه استفاده کنیم. آرایه‌ها لیستی از متغیرها می‌باشند که همگی از یک نوع هستند. برای مثال اگر بخواهید نام بعضی شهرها را نگهداری کنیم بایستی از یک آرایه از نوع رشته ای استفاده کنیم و یا برای نگهداری سن دوستانمان از آرایه صحیح استفاده می‌کنیم. در این فصل با انواع مختلف آرایه‌ها و نحوه ایجاد و طرز بکارگیری آن آشنا می‌شویم.

۲– تعریف و استفاده از آرایه :

هنگامی که آرایه تعریف می‌کنیم ، در واقع متغیری تعریف می‌کنیم که می‌تواند بیش از یک عنصر را در خود نگهداری کند. برای مثال اگر متغیر رشته‌ای به صورت زیر تعریف کنیم فقط می‌توانیم در هر زمان یک عنصر در آن جای دهیم :

```
String strName;
```

اما آرایه‌ها باعث می‌شوند که متغیر بیش از یک مقدار را در خود نگهداری کنند. برای اینکه بتوانیم یک آرایه را تعریف کنیم بایستی جلوی نوع متغیر از [] استفاده کنیم. طرز تعریف آرایه‌ها به صورت زیر است :

نوع آرایه [ ] نام‌آرایه = new نوع‌آرایه [اندازه‌آرایه];

برای مثال برای تعریف یک آرایه رشته‌ای با نام strName با اندازه 10 مکان به صورت زیر می‌نویسیم :

```
String[] strName = new String[10];
```

هنگامی که آرایه را تعریف کردیم ، می‌توانیم به تک‌تک عناصر آن با استفاده از اندیس آرایه مقدار داد.

مثلا برای اینکه به آرایه تعریف شده بالا در برنامه بخواهیم به مکانهای 0 و ۵ به ترتیب مقدار "Mehdi" و "Hadi" بدهیم به صورت زیر عمل می‌کنیم :

```
strName[0] = "Mehdi";
strName[5] = "Hadi";
```

نکته : اندیس آرایه‌ها از صفر شروع می‌شود. در مثال بالا به اندیس 5 مقدار "Hadi" دادیم یعنی به عنصر ششم آرایه مقدار دادیم.

نکته : برای دسترسی به مقدار آرایه نیز از همین اندیس استفاده می‌کنیم. مثلا برای نمایش عنصر ششم آرایه توسط تابع MessageBox به صورت زیر عمل می‌کنیم.

```
MessageBox.Show(strName[5]);
```

مثال۱ : برنامه‌ای بنویسید که یک آرایه رشته‌ای با 5 عنصر تعریف و مقادیر دلخواه را در هنگام Load شدن فرم بگیرد. (متغیر را به صورت عمومی تعریف کنید). این برنامه یک دکمه داشته باشد که با کلیک بر روی این دکمه تمامی عناصر آرایه را درون ListBox نمایش دهد.


```
String[] strFamily = new String[5];

private void Form1_Load(object sender, EventArgs e)
{
    strFamily[0] = "sobhkhiz";
    strFamily[1] = "Norouzi";
    strFamily[2] = "Azizi";
    strFamily[3] = "sedghi";
    strFamily[4] = "rayat parvar";
}

private void btnDisplay_Click(object sender, EventArgs e)
{
   foreach ( String x in strFamily )
      listBox1.Items.Add(x);
}
```

نکته : اگر می‌خواستیم آرایه از انتها به ابتدا درون ListBox نمایش داده شود ، بایستی از for به صورت زیر استفاده می‌کردیم.

```
for (int i = strFamily.Length - 1; i >= 0; i--)
    listBox1.Items.Add(strFamily[i]);
```

نکته : دستور Length اندازه آرایه را برمی‌گرداند و چون آرایه‌ها از صفر شروع می‌شوند ، در شروع اندیس i از اندازه آرایه یک مقدار کم کردیم تا به انتهای آرایه دسترسی داشته باشیم. و در حلقه for یکی یکی عناصر را نمایش می‌دهیم و هر بار از مقدار i یکی کم می‌کنیم.

۳– روش مقداردهی اولیه به آرایه :

بعضی مواقع ما مقادیری داریم که می‌خواهیم که هنگام تعریف آرایه این مقادیر را به آرایه نسبت دهیم. برای این کار می‌توانیم از آکولاد باز و بسته استفاده کنیم و مقادیر را به آرایه درون { } قرار دهیم. برای جدا کردن عنصر به عنصر از ویرگول استفاده می‌کنیم. برای مثال یک آرایه با اندازه ۶ تعریف می‌کنیم و ۶ مقدار به این آرایه نسبت می‌دهیم.

```
String[] strFamily = new String[6] { "mehdi", "hadi", "iman" , "ehsan", "morteza", "ebrahim" };
```

۴– آرایه‌های چند بعدی :

به آرایه‌ای که در بخش‌های قبلی استفاده کردیم آرایه یک بعدی گفته می‌شود. بعضی اوقات به آرایه‌ای بیش از یک بعد نیاز داریم. به آرایه یک بعدی ، لیست خطی و به آرایه دو بعدی ، جدول یا ماتریس و به آرایه سه بعدی ، فضا یا حجم گفته می‌شود. بیشترین کاربرد ارایه‌ها تا بعد سوم می‌باشد ، هر چند تا آرایه n بعدی هم می‌شود تعریف کرد. برای تعریف آرایه n بعدی به صورت زیر عمل می‌کنیم :

نوع آرایه [, , , ... ,] نام‌آرایه = new نوع‌آرایه [بعد۱ , بعد۲ , ... , بعد n];

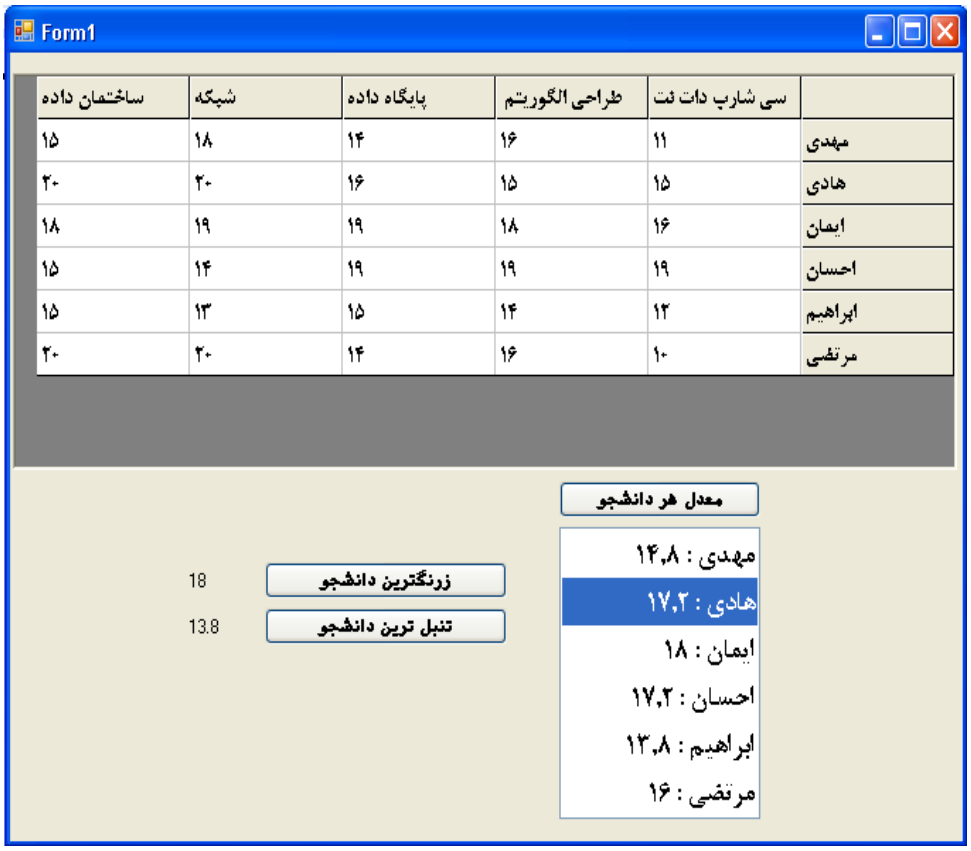
مثلا برای تعریف یک ماتریس که دارای ۳ سطر و ۵ ستون باشد که همگی مقادیر از نوع صحیح باشند به صورت زیر عمل کنید :

```
int[,] X = new int[3,5];
```

یا مثلا برای تعریف یک فضا که دارای ابعاد ۳×۴×۲ به صورت زیر عمل کنید :

```
int[,,] X = new int[2,4,3];
```

مثال۲ : میخواهیم برنامه ای بنویسیم که نمرات ۵ درس مربوط به ۶ دانشجو را درون آرایه‌ای قرار دهد ، نام درسها و نام دانشجویان نیز به طور جداگانه درون آرایه‌های دیگری قرار گیرد. برنامه برای نمایش اطلاعات دانشجویان از FlexGrid استفاده می‌کند. برنامه این قابلیت را دارد که درون listBox معدل تمامی دانشجویان را به همراه نام آنها نمایش می‌دهد. برنامه دو دکمه اجرای دیگری دارد که کار آنها نمایش معدل زرنگترین و تنبل‌ترین دانشجو می‌باشد (یادآوری : طرز اضافه کردن کنترل FlexGrid به فرم در فصل قبلی گفته شده است).




```
String[] strName = new String[6] { "مهدی","هادی","ایمان","احسان","ابراهیم","مرتضی" };
String[] strLesson = new String[5] { "نت دات شارپ سی", "الگوریتم طراحی ", "داده پایگاه",
                                     "شبکه", "داده ساختمان" };
Double[,] Grades = new Double[6, 5]  { { 11, 16, 14, 18, 15 },
                                       { 15, 15, 16, 20, 20 },
                                       { 16, 18, 19, 19, 18 },
                                       { 19, 19, 19, 14, 15 },
                                       { 12, 14, 15, 13, 15 },
                                       { 10, 16, 14, 20, 20 }};
```

```
private void Form1_Load(object sender, EventArgs e)
{
   MFG1.ROWS = 7;
   MFG1.COLS = 6;
   MFG1.FixedRows = MFG1.FixedCols = 1;
   MFG1.Col = 0;
   for (int i = 0; i < 6; i++)
   {
      MFG1.Row = i+1;
      MFG1.Text = strName[i];
   }
   MFG1.Row = 0;
   for (int i = 0; i < 5; i++)
   {
      MFG1.Col = i + 1;
      MFG1.Text = strLesson[i];
   }
   for (int i = 0; i < 6; i++)
   {
      MFG1.Row = i + 1;
      for (int j = 0; j < 5; j++)
      {
         MFG1.Col = j + 1;
         MFG1.Text = Grades[i,j].ToString();
      }
   }
}

private void btnAverage_Click(object sender, EventArgs e)
{
   for (int i = 0; i < 6; i++)
   {
      Double sum = 0;
      for (int j = 0; j < 5; j++)
      {
         sum = sum + Grades[i, j];
      }
      listBox1.Items.Add ( strName[i] + " : " + (sum/5).ToString("##.##"));
   }
}

private void btnZERANG_Click(object sender, EventArgs e)
{
   Double Max, sum = 0;
   for (int i = 0; i < 5; i++ )
      sum = sum + Grades[0,i];
   Max = (sum / 5.0);

   for (int i = 1; i < 6; i++)
   {
      sum = 0;
      for (int j = 0; j < 5; j++)
      {
         sum = sum + Grades[i, j];
      }
      if (Max < (sum) / 5.0)
         Max = (sum / 5.0);
   }
   label1.Text = Max.ToString("##.##");
}

private void btnTANBAL_Click(object sender, EventArgs e)
{
   Double Min, sum = 0;
   for (int i = 0; i < 5; i++)
      sum = sum + Grades[0, i];
   Min = (sum / 5.0);

   for (int i = 1; i < 6; i++)
   {
      sum = 0;
      for (int j = 0; j < 5; j++)
      {
         sum = sum + Grades[i, j];
      }
      if (Min > (sum) / 5.0)
        Min = (sum / 5.0);
   }
   label2.Text = Min.ToString("##.##");
}
```

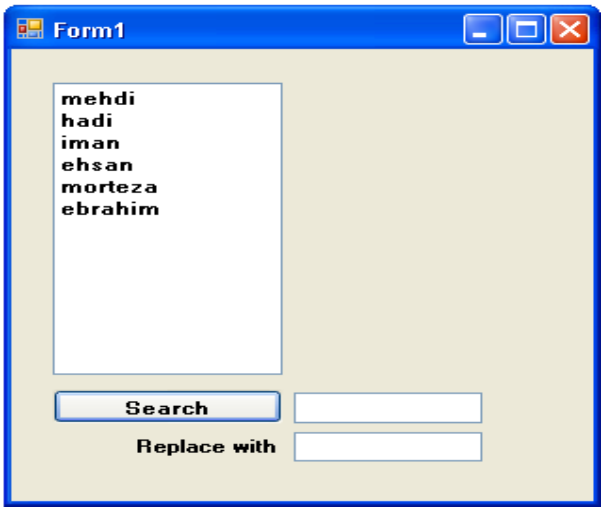
۵- جستجو در آرایه:

در بسیاری از اوقات میخواهیم عنصری را در آرایه جستجو کنیم. برای جستجو در آرایه می‌توانیم از حلقه for و دستور if به صورت زیراستفاده کنیم :

```
int index = -1;
for ( int i=0; i<آرایه.Length; i++ )
   if ( آرایه [i] == مقدار )
   {
      index = i;
      break;
   }
```

حال اگر بعد از اتمام حلقه for هنوز مقدار index برابر 1- باشد یعنی عنصر مورد نظر یافت نشد. در غیر اینصورت محتوای index برابر شماره مکان عنصر یافت شده درون آرایه خواهد شد و از index می‌توانیم برای پردازش روی آرایه مورد نظر استفاده کنیم.

مثال۳ : برنامه‌ای بنویسید که درون یک آرایه که نام ۶ نفر در آن موجود است یک نام دلخواه را جستجو کند و اگر نام مورد نظر پیدا شد به جای آن یک نام دلخواه دیگری قرار دهد.




```
String[] strName = new String[6] { "mehdi", "hadi", "iman", "ehsan", "morteza", "ebrahim" };

private void Form1_Load(object sender, EventArgs e)
{
    listBox1.Items.Clear();
    foreach (String x in strName)
        listBox1.Items.Add(x);
}

private void btnSearch_and_Replace_Click(object sender, EventArgs e)
{
    int index = -1;
    for ( int i=0; i<strName.Length; i++ )
       if (strName[i] == textBox1.Text)
       {
          index = i;
          break;
       }
    if (index == -1)
       MessageBox.Show("عنصر مورد نظر یافت نشد");
    else
    {
       strName[index] = textBox2.Text;
       Form1_Load(null, null);
    }
}
```

۶- مرتب سازی آرایه:

بعضی وقتها نیاز داریم که آرایه‌ای که داریم روی آن پردازش انجام می‌دهیم را به صورت صعودی یا نزولی مرتب کنیم. مرتب سازی به انواع مختلفی انجام می‌شود که بیشتر این مرتب سازی‌ها و الگوریتم‌های مرتب سازی در درس ساختمان داده‌ها مفصل بحث می‌شوند. ما یکی از الگوریتمهای مرتب سازی با نام تعویضی را در اینجا ذکر می‌کنیم :

در مثال قبلی آرایه‌ای با نام strName داشتیم حال می‌خواهیم یک دکمه به همان فرم اضافه کنیم و با زدن بر روی این دکمه عملیات مرتب سازی صعودی را انجام دهیم.

```
for ( int i=0; i<strName.Length; i++ )
   for (int j=0; j < strName.Length; j++)
      if ( String.Compare(strName[i],strName[j] ) < 0 )
      {
         String temp = strName[i];
         strName[i] = strName[j];
         strName[j] = temp;
      }
```

در خط سوم از یک تابع با نام Compare که جزء کلاس String می‌باشد استفاده کردیم. این تابع دو رشته را به عنوان پارامتر ورودی می‌پذیرد و براساس مقایسه دو رشته یک مقدار صحیح بر می‌گرداند که اگر این عدد صحیح صفر باشد دو رشته با هم برابرند ، اگر عدد منفی باشد رشته اول کوچکتر است و اگر عدد مثبت باشد رشته اول بزرگتر است.

در الگوریتم ارائه شده بالا مرتب سازی صعودی انجام می‌شود. اگر می‌خواهیم که مرتب سازی نزولی انجام شود از علامت < در خط سوم باید استفاده کنیم.

7– کلاس System.Array :

انواع آرایه‌ها از قبیل int[] ، double[] و غیره رفتارشان را از کلاسی به نام System.Array به ارث می‌برند. برای تعریف آرایه می‌توان به صورت زیر عمل کرد :

نوع‌آرایه نام‌آرایه = new نوع‌آرایه [تعداد عناصر] { مقادیر };

اگر از کلاس Array استفاده می‌کنید می‌توانید از متدهای این کلاس نیز استفاده کنید ، برخی از متدهای این کلاس عبارتند از :

متد	توضیحات
Copy	برای کپی کردن آرایه‌ای در آرایه دیگر به کار می‌رود. طرز کاربرد آن به صورت زیر است : `Array.Copy (آرایه مبدا , اندیس مبدا , آرایه مقصد , اندیس مقصد , طول کپی‌برداری )`
Sort	برای مرتب کردن عناصر آرایه به کار می‌رود. کاربرد آن به صورت زیر است : `Array.Sort ( آرایه );`
Reverse	برای معکوس کردن محتویات آرایه بکار می‌رود. کاربرد آن به صورت زیر است : `Array.Reverse ( آرایه );`

8– روش تولید اعداد تصادفی و ذخیره آن در آرایه :

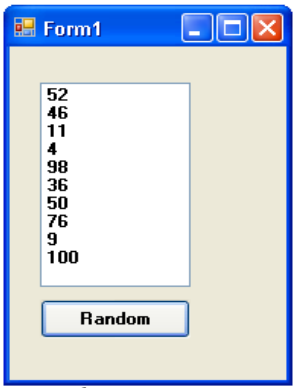
در بعضی مواقع نیاز داریم که اعداد تصادفی تولید کنیم و آنها را درون آرایه و یا هر چیز دیگری ذخیره کنیم. فرض کنید که در موسسه‌ای آزمونی برگزار خواهد شد که برگه امتحانی هر شخص با شخص دیگری متفاوت است. امتحان از طریق کامپیوتر انجام می‌شود. ما میخواهیم از بانک سئوالات شماره سئوالات را به صورت تصادفی انتخاب کنیم و برای هر شخص درون آرایه خودش نگهداری کنیم ، پس نیاز داریم تا با طرز تولید اعداد تصادفی آشنا شویم.

برای تولید اعداد تصادفی می‌توان از متغیری از نوع Random تعریف کرد :

```
Random نام متغیر  = new Random();
```

برای تولید اعداد تصادفی بعدی می‌توان از متد Next استفاده کرد. این متد می‌تواند یک پارامترعددی را نیز بگیرد که در این صورت عدد تصادفی تولید شده بین صفر تا آن پارامتر عددی تولید خواهد شد.

مثال۴ : برنامه‌ای بنویسید که ۱۰ عدد تصادفی بین ۱ تا ۱۰۰ را درون آرایه‌ای ذخیره کند و بعد درون یک جعبه‌لیست نمایش دهد.



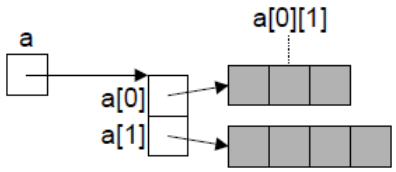

```
private void btnRANDOM_Click(object sender, EventArgs e)
{
   Random r1 = new Random();
   int[] a = new int[10];
   listBox1.Items.Clear();
   for (int i = 0; i < 10; i++)
   {
      a[i] = (r1.Next(100)) + 1;
      listBox1.Items.Add(a[i].ToString());
   }
}
```

۹– آرایه‌های دندانه‌دار (Jagged Array) :

آرایه‌هایی که تا به حال تعریف می‌کردیم (بیشتر از یک بعدی) برای هر بعد نمی‌توانستیم ، بعد دیگر آن را پویا در نظر بگیریم. برای آنکه بتوانیم بعدهای آرایه را پویا کنیم می‌توانیم از آرایه‌های دندانه‌دار استفاده کنیم. در شکل‌های زیر اختلاف بین آرایه‌های معمولی و آرایه‌های دندانه‌دار را می‌توانید متوجه شوید :

- **تعریف آرایه دندانه‌دار**

```
int[][] a = new int[2][];
a[0] = new int[3];
a[1] = new int[4];
```

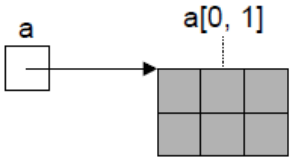


– طرز مقدار دادن به آرایه دندانه‌دار

```
a[i][j] = مقدار ;
```

- **تعریف آرایه معمولی**

```
int[,] a = new int[2,3];
```

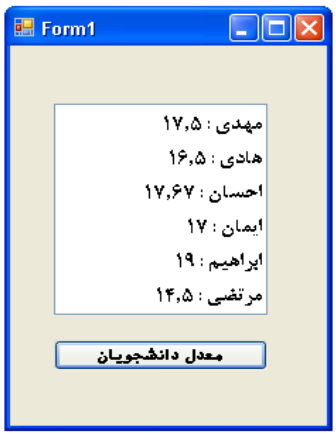


– طرز مقدار دادن به آرایه دندانه‌دار

```
a[i,j] = مقدار ;
```

مزیت آرایه‌های دندانه‌دار نسبت به آرایه‌های معمولی این است که به اندازه مورد نیاز ما حافظه از سیستم برای ذخیره مقادیر مورد نیاز می‌گیریم.

مثال۵ : برنامه‌ای بنویسید که طبق جدول زیر نمرات درسی دانشجویان را درون آرایه ذخیره کند. سپس معدل هر دانشجو را درون جعبه‌لیست ذخیره کند.



مهدی	۲۰	۱۵	۱۸	۱۷	–
هادی	۱۴	۱۹	–	–	–
ایمان	۱۶	۱۷	۲۰	–	–
احسان	۱۳	۱۹	۲۰	۱۶	۱۷
ابراهیم	۱۹	–	–	–	–
مرتضی	۱۳	۱۴	۱۵	۱۶	–

```
private void btnAverage_Click(object sender, EventArgs e)
{
  String[] strName = new String[6] { "مهدی", "هادی", "احسان", "ایمان", "ابراهیم", "مرتضی" };
  int[][] grades = new int[6][];
  grades[0] = new int[4] { 20, 15, 18, 17 };
  grades[1] = new int[2] { 14, 19 };
  grades[2] = new int[3] { 16, 17, 20 };
  grades[3] = new int[5] { 13, 19, 20, 16, 17 };
  grades[4] = new int[1] { 19 };
  grades[5] = new int[4] { 13, 14, 15, 16 };
  for (int i = 0; i < 6; i++)
  {
     int sum = 0;
     for (int j = 0; j < grades[i].Length; j++)
         sum = sum + grades[i][j];
     listBox1.Items.Add(strName[i] + " : " + (sum*1.0 / grades[i].Length).ToString("##.۰۰"));
  }
}
```

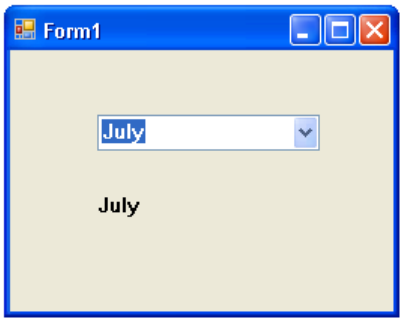
۱۰– شمارنده‌ها

متغیرهایی که تاکنون استفاده کردیم هیچ محدودیتی در نوع اطلاعاتی که می‌توانستند در خود ذخیره کنند نداشتند. برای مثال اگر متغیری از نوع int تعریف می‌کردید ، می‌توانستید هر عدد صحیحی را در آن ذخیره کنید. این مساله برای متغیرهای از نوع صحیح هم وجود داشت. اما در بعضی شرایط ممکن است بخواهید که متغیر شما اعداد محدودی را در خود نگهداری کند. برای مثال فرض کنید می‌خواهید متغیر صحیحی تعریف کنید که تعداد درهای اتومبیل را در خود نگهداری کند ، قطعا نمی‌خواهید که این متغیر عدد بزرگی مثل ۱۶۳۲۷ را در خود ذخیره کند. برای رفع این مشکل می‌توانید از شمارنده‌ها استفاده کنید.

با استفاده از شمارنده‌ها می‌توانید انواع داده‌ای جدیدی از انواع موجود از قبیل int و Double و String و غیره تعریف کنید. متغیرهایی که از این نوع داده‌ای جدید ایجاد می‌شوند ، فقط می‌توانند مقداری را داشته باشند که شما مشخص می‌کنید. به این ترتیب می‌توانید در برنامه از وارد شدن اعداد غیر منطقی در متغیرها جلوگیری کنید. همچنین استفاده از شمارنده‌ها در کد برنامه باعث افزایش خوانایی برنامه شما می‌شود.

طرز تعریف و کار با شمارنده‌ها را با یک مثال توضیح می‌دهیم.

مثال۶: برنامه‌ای بنویسید که نام ماههای سال را درون یک comboBox قرار دهد و اگر کاربر هر کدام از ماهها را انتخاب کرد ، نام آن ماه درون یک برچسب نمایش داده شود.




```
private enum MONTH_Enum
{
    January = 0,
    February,
    March,
    April,
    May,
    June,
    July,
    August,
    September,
    October,
    November,
    December
 };
```

نام شمارنده

نکته۱: برنامه خود برای شمارنده ، شماره‌ها را از 0 شروع می‌کند و ما می‌توانستیم که 0 را برای متغیر اولی ننویسیم.

نکته۲ : اگر متغیر اولی را مقدار خاصی بدهیم ولی بقیه را مقدار ندهیم ، شماره بقیه متغیرها به ترتیب از مقدار یک واحد افزایشی مقدار خاص بدست می‌آیند.

نکته۳ : برای هر کدام از متغیرها می‌توانیم شماره خاصی در نظر بگیریم. شماره‌ها نباید تکراری باشند.

```
private void Form1_Load(object sender, EventArgs e)
{
    comboBox1.Items.Add(MONTH_Enum.January);
    comboBox1.Items.Add(MONTH_Enum.February);
    comboBox1.Items.Add(MONTH_Enum.March);
    comboBox1.Items.Add(MONTH_Enum.April);
    comboBox1.Items.Add(MONTH_Enum.May);
    comboBox1.Items.Add(MONTH_Enum.June);
    comboBox1.Items.Add(MONTH_Enum.July);
    comboBox1.Items.Add(MONTH_Enum.August);
    comboBox1.Items.Add(MONTH_Enum.September);
    comboBox1.Items.Add(MONTH_Enum.October);
    comboBox1.Items.Add(MONTH_Enum.November);
    comboBox1.Items.Add(MONTH_Enum.December);
    comboBox1.Text = MONTH_Enum.January.ToString();
}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
     switch (comboBox1.SelectedIndex)
     {
        case 0:  label1.Text = MONTH_Enum.January.ToString();     break;
        case 1:  label1.Text = MONTH_Enum.February.ToString();    break;
        case 2:  label1.Text = MONTH_Enum.March.ToString();       break;
        case 3:  label1.Text = MONTH_Enum.April.ToString();       break;
        case 4:  label1.Text = MONTH_Enum.May.ToString();         break;
        case 5:  label1.Text = MONTH_Enum.June.ToString();        break;
        case 6:  label1.Text = MONTH_Enum.July.ToString();        break;
        case 7:  label1.Text = MONTH_Enum.August.ToString();      break;
        case 8:  label1.Text = MONTH_Enum.September.ToString();   break;
        case 9:  label1.Text = MONTH_Enum.October.ToString();     break;
        case 10: label1.Text = MONTH_Enum.November.ToString();    break;
        case 11: label1.Text = MONTH_Enum.December.ToString();    break;
      }
}
```

۱۱- ساختارها

هنگام نوشتن نرم‌افزار در بیشتر مواقع به اطلاعاتی نیاز دارید که مقادیر مختلفی را در خود نگه داشته باشد. در اینگونه موارد بایستی از کلاس یا ساختار استفاده کرد. کلاسها را در فصل‌های بعدی توضیح می‌دهیم. طرز تعریف ساختار به صورت زیر است :

```
public  struct  نام‌ساختار
{
        public  نوع‌متغیرها  نام‌متغیرها;
}
```

برای مثال ساختار یا رکورد دانشجویی را با مشخصات زیر تعریف می‌کنیم :

نام فیلد	نوع فیلد	توضیحات
stno	Int32	شماره دانشجویی
name	String	نام
family	String	نام خانوادگی
email	String	ایمیل
lessons	String[]	نام درسها
grades	Double[]	نمره درسها

توجه کنید که lessons و grades به صورت آرایه‌ای تعریف شده‌اند.

```
public struct Student
{
    public Int32 stno;
    public String name;
    public String family;
    public String Email;
    public String[] lessons;
    public Double[] grades;
};
```

برای اینکه از این ساختار در برنامه استفاده کنیم بایستی به صورت زیر متغیری از نوع رکورد تعریف شده بسازیم و از این متغیر استفاده کنیم :

```
Student st;
```

حال با استفاده از این متغیر و نقطه می‌توانیم به فیلدهای مورد نظر تعریف شده دستیابی داشته باشیم ، مثلا برای اینکه به فیلد family دسترسی داشته باشیم به صورت زیر عمل می‌کنیم :

```
st.family = "sobhkhiz";
```

برای اینکه به فیلدهای آرایه‌ای دسترسی داشته باشیم بایستی فیلدهای آرایه‌ای را بسازیم و از آن استفاده کنیم. برای مثال اگر بخواهیم ۵ نام درس و ۵ نمره درسی دانشجوی st را مقدار دهی کنیم به صورت زیر عمل می‌کنیم :

```
st.lessons = new String[5] { "C#", "Database", "Network", "C++", "Pascal" };
st.grades = new Double[5] { 15, 16, 17, 18, 19 };
```

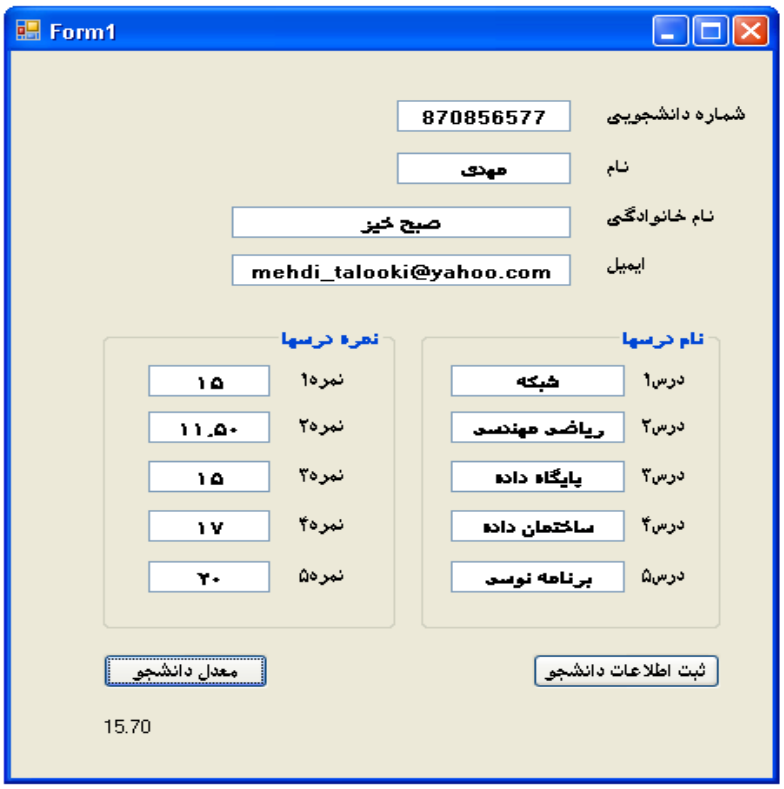
مثال۷: برنامه‌ای بنویسید که رکورد دانشجوی بالا را تعریف کند و از آن در برنامه استفاده کنید. برنامه قابلیت گرفتن معدل‌گیری برای دانشجوی مورد نظر را داشته باشد. (کنترل‌های مورد نظر را بر اساس فیلدها مشخص کنید).

```
public struct Student
{
   public Int32 stno;
   public String name;
   public String family;
   public String Email;
   public String[] lessons;
   public Double[] grades;
};
Student st;
private void btnSABT_Click(object sender, EventArgs e)
{
   st.stno = Convert.ToInt32(textBox1.Text.Trim());
   st.name = textBox2.Text;
   st.family = textBox3.Text;
   st.Email = textBox4.Text;
```

نکته: Trim() فاصله‌های خالی دو طرف رشته را حذف می‌کند.

```
    st.lessons = new String[5];
    st.lessons[0] = textBox5.Text;
    st.lessons[1] = textBox6.Text;
    st.lessons[2] = textBox7.Text;
    st.lessons[3] = textBox8.Text;
    st.lessons[4] = textBox9.Text;
    st.grades = new Double[5];
    st.grades[0] = Convert.ToDouble(textBox10.Text.Trim());
    st.grades[1] = Convert.ToDouble(textBox11.Text.Trim());
    st.grades[2] = Convert.ToDouble(textBox12.Text.Trim());
    st.grades[3] = Convert.ToDouble(textBox13.Text.Trim());
    st.grades[4] = Convert.ToDouble(textBox14.Text.Trim());
}

private void btnAverage_Click(object sender, EventArgs e)
{
    Double sum = 0;
    for (int i = 0; i < 5; i++)
       sum = sum + st.grades[i];
    lblAverage.Text = (sum / 5).ToString("##.00");
 }
```



۱۲– تمرین در خانه

۱– برنامه مثال ۲ را با استفاده از ساختار بنویسید. در برنامه بایستی از آرایه از رکوردها به صورت زیر استفاده کنید:

```
Student[] st = new Studen[6];
```

۲– برنامه مثال ۲ را با ساختار و طوری بنویسید که نمرات را از `DataGridView` بخواند. (از `FlexGrid` استفاده نکنید)

توضیحات :

- برنامه‌ها را به سلیقه خودتان بنویسید
- برای دسترسی به فیلد نام دانشجوی دوم بایستی بنویسیم ← `st[1].name`
- برای دسترسی به فیلد نمره درس چهارم از دانشجوی اول بایستی بنویسیم ← `st[0].grades[3]`

فصل ششم : توابع و کلاس‌ها

۱- توابع و زیربرنامه‌ها
۲- نوشتن توابع
۳- روشهای ارسال پارامترها به توابع
۴- ارسال آرایه به توابع
۵- توابع و متدهای همنام
۶- برخی از توابع ریاضی
۷- کلاس‌ها
۸- تعریف کلاس
۹- سازنده‌ها
۱۰-تمرین در خانه

۱- توابع و زیربرنامه‌ها

در برنامه‌های بزرگ ، برنامه را به قطعات کوچکتر تقسیم می‌کنند تا حل برنامه‌های بزرگ آسان شود. هر یک از این قطعات را می‌توان به صورت یک برنامه نوشت که آنها را تابع یا زیربرنامه می‌گویند. بنابراین تابع برنامه‌ای است که برای حل بخشی از مسئله به کار می‌آید در برنامه‌نویسی ساختیافته توابع به صورت مستقل عمل می‌کنند. ولی در برنامه‌های شیءگرا توابع به عنوان عضو کلاس در نظر گرفته می‌شود. در زیر مزیت‌های توابع آورده شده است :

- توابع موجب آسان‌تر شدن حل مساله‌های بزرگ می‌شود.
- همکاری بین افراد مختلفی که دارند روی برنامه بزرگ کار می‌کنند را فراهم می‌کنند به طوری که هر فرد بخش کوچکی از مساله را حل کند.
- اشکال زدایی برنامه آسان می‌شود.
- توابع از تکرار دستورات جلوگیری می‌کنند. یعنی یک تابع را یک بار می‌نویسیم و چندبار آن را فراخوانی می‌کنیم.

۲- نوشتن تابع

برای نوشتن هر تابع باید اهداف آن مشخص باشد. یعنی تابع چه وظایفی بر عهده دارد ، ورودیهای تابع چیست و خروجی‌های آن کدام است. با دانستن این موارد نوشتن توابع کار دشواری نیست. تابع در C#.Net دارای دو جنبه است :

- تعریف تابع
- جنبه فراخوانی

فراخوانی تابع با نام آن انجام می‌شود. چنانچه تابع دارای پارامتر باشد ، پارامترها نیز هنگام فراخوانی به تابع ارسال می‌شوند. در زیر طریقه نوشتن تابع آورده شده است.

```
[ private | public ] نوع‌تابع    نام‌تابع    (    لیست پارامترها    )
{
        دستورات تابع
}
```

در تعریف تابع که در بالا ذکر شده است هر تابع می‌تواند یک مقدار را برگشت دهد اگر نوع تابع را غیر از void در نظر بگیریم بایستی برای برگشت دادن یک مقدار از دستور return استفاده کنیم.

برای مثال تابعی تعریف کنید که دو پارامتر به عنوان ورودی داشته باشد و کار تابع بدین صورت باشد که تابع مقدار ماکزیمم را برگشت دهد.

```
int findmax ( int x, int y )
{
   if ( x >= y )
     return x;
   return y;
}
```

دستور return بسیار قوی است و طوری تعریف شده است که به محض اجرای این دستور ، کنترل برنامه از تابع خارج می‌شود و به خط بعدی تابع فراخوان برمی‌گردد. طرز بکارگیری تابع فراخوان تابع بالا به صورت زیر است.

```
int m = findmax ( 15,16 );
```

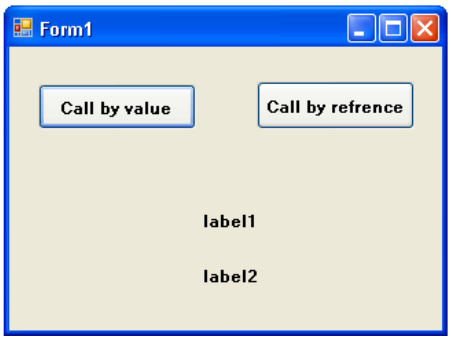
چون مقداری که تابع بر می‌گرداند از نوع صحیح است پس سمت راست تابع فراخوان بایستی یک نوع متغیر صحیح تعریف کنیم. در ضمن قانون نامگذاری تابع از قانون نام گذاری متغیرها پیروی می‌کند.

۳- روشهای ارسال پارامترها به توابع

توابع می‌توانند فاقد پارامتر باشند ، اما اگر توابع دارای پارامتر باشند ، پارامتر می‌تواند به سه روش ارسال شود :

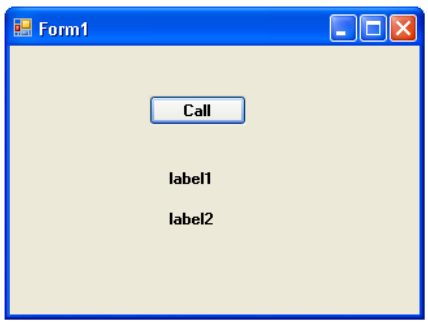
- **فراخوانی با مقدار (**Call by value**)** : در فراخوانی با مقدار ، در هنگام فراخوانی ، پارامترهای واقعی در پارامترهای مجازی کپی می‌شود. از آن لحظه به بعد ، هر تغییر در پارامترهای مجازی تاثیری در پارامترهای واقعی ندارد.
- **فراخوانی با ارجاع (**Call by refrence**)** : برخلاف فراخوانی با مقدار هر تغییری در پارامترهای مجازی بر روی پارامترهای واقعی تاثیرخواهد گذاشت. برای فراخوانی با ارجاع از کلمه کلیدی ref استفاده می‌شود. قبل از استفاده از پارامتری که به صورت ref ارسال می‌شود بایستی قبل از ارسال مقدار اولیه بگیرد.
- **استفاده از پارامترهای Out :** در فراخوانی با ارجاع اگر پارامتری قبل از ارسال مقدار اولیه نگیرد با خطا مواجه خواهیم شد. برای رفع این مشکل می‌توانیم قبل از ارسال پارامتر از کلمه کلیدی out استفاده کنیم.

مثال۱ : برنامه‌ای می‌نویسیم که تفاوت فراخوانی با مقدار و فراخوانی با مقدار را نمایش می‌دهد :




```
private void test1(int x, int y)
{
   x++;
   y++;
}
private void test2(ref int x, ref int y)
{
   x++;
   y++;
}
private void btnCallByValue_Click(object sender, EventArgs e)
{
   int x = 50;
   int y = 100;
   test1(x, y);
   label1.Text = x.ToString();
   label2.Text = y.ToString();
}
private void btnCallByRefrence_Click(object sender, EventArgs e)
{
   int x = 50;
   int y = 100;
   test2(ref x, ref y);
   label1.Text = x.ToString();
   label2.Text = y.ToString();
}
```

مثال۲ : : برنامه‌ای می‌نویسیم که تفاوت فراخوانی با ارسال ref و out را نمایش می‌دهد.



ابتدا کد فرم بالا را به صورت زیر بنویسید و اجرا کنید :

```
private void test (ref int x, ref int y)
{
   x = 50;
   y = 100;
}

private void button1_Click(object sender, EventArgs e)
{
   int x;
   int y;
   test(ref x, ref y);
   label1.Text = x.ToString();
   label2.Text = y.ToString();
}
```

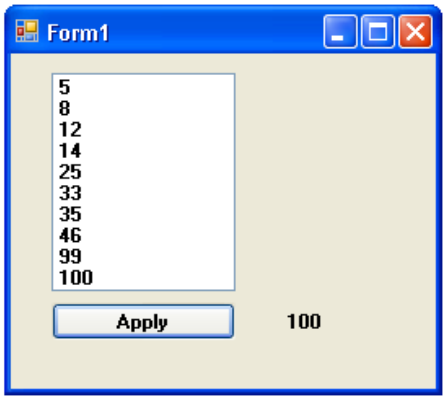
چون قبل از ارسال از طریق ref به پارامترهای x و y مقدار اولیه ندادیم برنامه با خطا مواجه می‌شود. برای رفع این مشکل می‌توانیم از out به جای ref استفاده کنیم و یا به متغیرهای x و y مقدار اولیه‌ای مثال صفر بدهیم:

```
int x = 0;
int y = 0;
```

۴- ارسال آرایه به توابع

آرایه را می‌توان به صورت معمولی به توابع یا متد فراخوان ارسال کرد. در این صورت مقادیر پارامتر واقعی در پارامتر مجازی کپی و هر تغییری در پارامتر مجازی تاثیر خود را بر روی پارامتر واقعی می‌گذارد

مثال۳ : برنامه‌ای بنویسید که آرایه‌ای صحیح با مقادیر دلخواه را به تابعی ارسال می‌کند و این تابع آرایه را مرتب می‌کند ، همچنین این تابع بزرگترین مقدار آرایه را بر می‌گرداند.




```
private int Function(int[] my_array)
{
   for ( int i=0; i<my_array.Length; i++ )
     for (int j = 0; j < my_array.Length; j++)
       if (my_array[i] < my_array[j])
       {
          int temp = my_array[i];
          my_array[i] = my_array[j];
          my_array[j] = temp;
       }
   return my_array[my_array.Length - 1];       //آخرین عنصر آرایه مرتب شده را برمیگرداند
}

private void button1_Click(object sender, EventArgs e)
{
   int[] x = new int[10] { 5, 8, 99, 14, 100, 25, 46, 35, 33, 12 };
   int max = Function(x);
   listBox1.Items.Clear();
   foreach (int c in x)
     listBox1.Items.Add(c.ToString());
   label1.Text = max.ToString();
}
```

۵- توابع و متدهای همنام

توابع می‌توانند همنام باشند. برای اینکه توابع همنام را از هم تشخیص داده شوند ، باید از لیست پارامترها آن‌ها را تشخیص داد. به عنوان مثال توابع زیر را درنظر بگیرید :

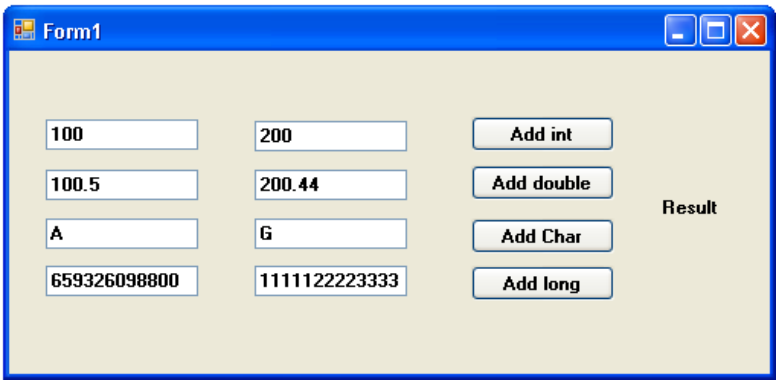
```
private void test(int x, int y);

private void test(double x);

private void test(double x, double y);
```

در سه تابع بالا از یک نام برای هر سه تابع استفاده شده است. طرز شناسایی تابع را تابع فراخواننده انجام می‌دهد. برای مثال اگر تابع فراخواننده دو پارامتر صحیح ارسال کند یعنی اینکه تابع خط اول صدا زده شده است و اگر دو پارمتر از نوع اعشاری ارسال کند یعنی اینکه تابع خط سوم صدا زده شده است.

مثال۴ : برنامه‌ای می‌نویسیم که در آن طرز کار توابع همنام را نشان می‌دهد.




```
private int Add(int x, int y) {  return (x + y);  }

private Double Add(Double x, Double y)  {  return (x + y);  }

private int Add(Char x, Char y) {  return (x + y);  }

private long Add( long x, long y) {  return (x + y);  }


private void button1_Click(object sender, EventArgs e)
{
    int x = Convert.ToInt32(textBox1.Text);
    int y = Convert.ToInt32(textBox2.Text);
    lblResult.Text = Add(x, y).ToString();
}

private void button2_Click(object sender, EventArgs e)
{
     Double x = Convert.ToDouble(textBox3.Text);
     Double y = Convert.ToDouble(textBox4.Text);
     lblResult.Text = Add(x, y).ToString("##.000");
}

private void button3_Click(object sender, EventArgs e)
{
     Char x = Convert.ToChar(textBox5.Text);
     Char y = Convert.ToChar(textBox6.Text);
     lblResult.Text = Add(x, y).ToString();
}

private void button4_Click(object sender, EventArgs e)
{
     long x = Convert.ToInt64(textBox7.Text);
     long y = Convert.ToInt64(textBox8.Text);
     lblResult.Text = Add(x, y).ToString();
}
```

۶- برخی توابع ریاضی

بسیاری از توابع ریاضی وجود دارند که برای پیاده‌سازی این توابع ریاضی که بسیار هم دقیق باید باشند ما دچار مشکل هستیم. برخی از توابع ریاضی در زبان سی‌شارپ وجود دارند که کار ما را در این زمینه راحت می‌کنند. این توابع از کلاس Math ناشی می‌شوند. در جدول زیر برخی از توابع مهم این کلاس به همراه توضیح آنها آورده شده است.

برخی از توابع کلاس Math

تابع	توضیحات
Abs(x)	قدر مطلق یک داده عددی را محاسبه می‌کند.
Asin (x)	آرک سینوس یک داده عددی را محاسبه می‌کند
Acos (x)	آرک کسینوس یک داده عددی را محاسبه می‌کند
Atan (x)	آرک تانژانت یک داده عددی را محاسبه می‌کند
Atan2(x , y)	این تابع x را بر y تقسیم میکند و آرک تانژانت حاصل را محاسبه می‌کند.
BigMul (x, y)	این تابع برای ضرب اعداد بزرگ پیاده سازی شده است
Ceiling (x)	این تابع سقف یک مقدار اعشاری یا بزرگترین عدد صحیح یک مقدار اعشاری را بدست می‌آورد. Math.Ceiling (5.01) = 6

Cos (x)	کسینوس زوایه را بدست می‌آورد (البته بایستی عدد را به زاویه تبدیل کرد که گفته می‌شود)
Cosh (x)	کسینوس هایپربولیک را بدست می‌آورد
Exp (x)	e^x را محاسبه می‌کند.
Floor (x)	این تابع کف یک مقدار اعشاری یا کوچکترین عدد صحیح یک مقدار اعشاری را بدست می‌آورد. Math.Ceiling (5.99) = 5
Log (x, a)	این تابع لگاریتم x در مبنای a را محاسبه می‌کند.
Log10 (x)	این تابع لگاریتم در مبنای 10 را محاسبه می‌کند.
Max (x, y)	مقدار ماکزیمم بین x و y را محاسبه می‌کند.
Min (x, y)	مقدار مینیمم بین x و y را محاسبه می‌کند.
Pow (x, y)	x به نمای y را محاسبه می‌کند. (توان)
Round (x)	گرد شده عدد x را محاسبه می‌کند.
Sign (x)	علامت داده عددی x را بدست می‌آورد. اگر x مثبت باشد 1 و اگر x صفر باشد 0 و اگر منفی باشد 1- برمی‌گرداند.
Sin (x)	سینوس زوایه را بدست می‌آورد (البته بایستی عدد را به زاویه تبدیل کرد که گفته می‌شود)
Sinh (x)	سینوس هایپربولیک را بدست می‌آورد
Sqrt (x, y)	جذر را محاسبه می‌کند. x عدد داخل رادیکال و y نما می‌باشد.
Tan (x)	تانژانت زاویه را بدست می‌آورد.
Tanh (x)	تانژانت هایپربولیک را بدست می‌آورد.
Trancate (x)	مقدار اعشاری یک داده اعشاری را حذف می‌کند و داده صحیح آن را برمی‌گرداند.

دو ثابت درون کلاس Math قرار دارند به نامهای PI و E که اولی عدد پی و دومی عدد نپر می‌باشد.

نکته۱ : برای بدست آوردن عدد پی در هر زبان برنامه‌نویسی اگر ثابت PI را نداریم می‌توانیم از آرک تانژانت استفاده کنیم. فرمول آن به صورت زیر است :

```
Double Pi = Math.Atan(1) * 4;
```

نکته۲ : برای تبدیل یک عدد به زاویه از فرمول زیر استفاده کنید :

```
int x = 45;
Double grade = (x * Math.PI / 180);
```

۷– کلاس‌ها

کلاس‌ها برای مدلسازی اشیای دنیای واقعی بکار می‌روند. اجزایی که قسمتهای مختلف یک کلاس را می‌سازند عبارتند از : **صفات** و **رفتارها**. صفات را به صورت متغیر تعریف می‌کنیم که اعضای داده‌ای نیز نامیده می‌شوند. رفتارها را به صورت توابع عضو پیاده سازی کرده که متد نیز نامیده می‌شوند. این توابع عضو عملیاتهایی هستند که بر روی اعضای داده‌ای اجرا می‌شوند.

۸– تعریف کلاس

حال که با تعریف کلاس آشنا شدیم و فهمیدیم که برنامه سی‌شارپ از کلاس‌ها تشکیل می‌شود ، چگونگی تعریف کلاس را می‌آموزیم :

```
class نام‌کلاس
{
        نوع‌دسترسی نوع‌داده نام‌داده‌ها ;
        نوع‌دسترسی نوع‌برگشتی نام‌تابع ( لیست‌پارامترها )
        {
                دستورات تابع
        }
}
```

نوع دسترسی به در زبان C# به ۵ دسته تقسیم می‌شود : **اختصاصی – عمومی – محافظت‌شده – داخلی – داخلی‌محافظت‌شده**

در اینجا ما فقط دو دسته اول را بررسی می‌کنیم :

نوع‌اختصاصی : اعضا و انواع اختصاصی آنهایی هستند که فقط در داخل کلاسی که تعریف می‌شوند ، قابل استفاده‌اند. این اعضا و نوع ، توسط کلمه کلید private تعریف می‌شوند.

نوع عمومی : اعضای عمومی ، آنهایی هستند که در سایر کلاسهای موجود در برنامه قابل استفاده‌اند. اعضای عمومی پس از کلمه کلیدی public قرار می‌گیرند.

مثال : کلاس مستطیل را با هم پیاده‌سازی می‌کنیم. نام کلاس را Rectangle می‌گیریم. (این کلاس را به دو صورت داده اختصاصی و داده عمومی می‌نویسیم تا فرق بین این دو مورد را متوجه شویم)

اعضای داده‌ای مستطیل عبارتند از : طول – عرض

توابع عملیاتی عبارتند از :

- مقدار دادن به طول و عرض
- محاسبه مساحت
- محاسبه محیط

برای تعریف کلاس از منوی Project گزینه Add Class را کلیک کنید و از کادری که باز می‌شود نام کلاس که Rectangle می‌باشد را وارد کنید.

صورت عمومی اعضای داده‌ای : اگر این اعضا به صورت اختصاصی تعریف شوند ، فقط خود کلاس می‌تواند به صورت مستقیم به این اعضا دست داشته باشد. و در برنامه‌های دیگر قابل استفاده نیستند. برای دسترسی اعضای داده‌ای اختصاصی بایستی یک سری توابع عملیاتی نوشت که در مثال‌های بعدی خواهیم گفت.

```
class Rectangle
{
   public int len, wid;

   public int area()
   {
      return len * wid;
   }

   public int prime()
   {
      return 2 * (len + wid);
   }
}
```

حال فرم برنامه‌تان را به صورت زیر طراحی کنید و کد مورد نظر را بنویسید :


```
private void button1_Click(object sender, EventArgs e)
{
    Rectangle r1 = new Rectangle();
    r1.len = Convert.ToInt32(textBox1.Text);
    r1.wid = Convert.ToInt32(textBox2.Text);
    lblArea.Text = r1.area().ToString();
    lblPrime.Text = r1.prime().ToString();
}
```

نکته۱ : هر شی در زبان C# با استفاده از کلمه کلیدی new تعریف میشود. مثلا در مثال بالا نوشتیم Rectangle r1 = new Rectangle()

نکته۲: برای دسترسی به اعضای داده‌ای و توابع عملیاتی هر شی ایجاد شده از نقطه استفاده می‌کنیم. مثلا : r1.area(); یا r1.set_data(x,y);

صورت اختصاصی اعضای داده‌ای: در کلاس مورد نظر چون ما اعضای داده‌ای len و wid را به صورت عمومی تعریف کردیم در کد دکمه "محاسبه مساحت و محیط" توانستیم با r1 که یک شی از کلاس Rectangle می‌باشد به صورت r1.len و r1.wid به اعضای داده‌ای دسترسی داشته باشیم. اگر در کلاس مورد نظر اعضای داده‌ای len و wid را به صورت اختصاصی تعریف کنیم دیگر اجازه همچنین کاری به ما داده نخواهد شد. برای رفع این مشکل ، یعنی برای دسترسی به اعضای داده‌ای اختصاصی بایستی در کلاس مورد نظر توابعی تعریف کنیم.

کلاس مورد نظر بالا را به صورت زیر تغییر دهید :

```
class Rectangle
{
   private int len, wid;

   public void set_data(int x, int y)
   {
      len = x;
      wid = y;
   }

   public int area()
   {
      return len * wid;
   }

   public int prime()
   {
      return 2 * (len + wid);
   }
}
```

در این کلاس یک تابع عمومی با نام set_data تعریف کردیم (نام این تابع دلخواه است) که دو آرگومان ورودی دارد که اعضای داده‌ای کلاس با آن تنظیم می‌شوند.

کد دکمه "محاسبه مساحت و محیط" را به صورت زیر تغییر دهید.

```
private void button1_Click(object sender, EventArgs e)
{
    Rectangle r1 = new Rectangle();
    int x = Convert.ToInt32(textBox1.Text);
    int y = Convert.ToInt32(textBox2.Text);
    r1.set_data(x, y);
    lblArea.Text = r1.area().ToString();
    lblPrime.Text = r1.prime().ToString();
}
```

حال اگر در برنامه بخواهیم در برنامه‌مان به مقادیر اختصاصی دستیابی داشته باشیم ، برای مثال اگر بخواهیم مقدار طول و عرض را چاپ کنیم. بایستی تابعی بنویسیم که برای ما طول و عرض مستطیل را برگشت دهد. برای اینکه در مثال بالا مقدار طول و عرض مستطیل را بخوانیم دو تابع جدید زیر را به کلاس اضافه کنید (نام این دو تابع نیز دلخواه است):

```
public int get_len()
{
      return len;
}

public int get_wid()
{
      return wid;
}
```

حال در برنامه اگر بخواهیم مقدار طول و عرض مستطیل را درون دو برچسب نمایش دهیم به صورت زیر می‌نویسیم :

```
label1.Text = r1.get_len().ToString();
label2.Text = r1.get_wid().ToString();
```

تا اینجا با مفهوم داده‌های اختصاصی و داده‌های عمومی آشنا شدیم.

نکته۱: هر شی در زبان C# با استفاده از کلمه کلیدی new تعریف میشود. مثلا در مثال بالا نوشتیم `Rectangle r1 = new Rectangle()`

نکته۲: برای دسترسی به اعضای داده‌ای و توابع عملیاتی هر شی ایجاد شده از نقطه استفاده می‌کنیم. مثلا : `r1.area();` یا `r1.set_data(x, y);`

۹– سازنده‌ها

وقتی ما شی از نوع کلاس تعریف شده با دستور new ایجاد می‌کنیم در واقع سازنده کلاس را فراخوانی می‌کنیم. ما در تعریف کلاس می‌توانیم سازنده‌هایی را ایجاد کنیم و با این سازنده‌ها به اعضای داده‌ای موجود مقدار اولیه دهیم بکار می‌روند. سازنده‌ها همنام با نام کلاس در خود کلاس تعریف می‌شوند و هیچ نوعی را هم برنمی‌گردانند و حتی از نوع void هم نیستند. در زیر سه سازنده به کلاس Rectangle اضافه می‌کنیم. تا با مفهوم سازنده آشنا شویم :

```
class Rectangle
{
   private int len, wid;
   public Rectangle()
   {
      len = 0;
      wid = 0;
   }

   public Rectangle(int x)
   {
      len = x;
      wid = 0;
   }

   public Rectangle(int x, int y)
   {
      len = x;
      wid = y;
   }
  .
  .
  .
}
```

سازنده‌ای بدون پارامتر

سازنده‌ای با یک پارامتر

سازنده‌ای با دو پارامتر

حال در برنامه اگر بخواهیم به صورتهای زیر شی r1 را بسازیم داریم :

```
Rectangle r1 = new Rectangle();          //➔   r1.len = 0, r1.wid = 0
Rectangle r1 = new Rectangle( 5 );       //➔   r1.len = 5, r1.wid = 0
Rectangle r1 = new Rectangle( 5, 8 );    //➔   r1.len = 5, r1.wid = 8
```

تا اینجا توانستیم کلاس را به طور مختصر توضیح دهیم. هر چند خود کلاس‌ها به اندازه یک کتاب مطلب دارد. ولی تا همیجا ما به آن نیاز داریم.

مثال : کلاس اعداد گویا را می‌خواهیم با هم پیاده‌سازی کنیم :

اعداد گویا از دو قسمت صورت و مخرج تشکیل شده‌اند. عملیاتهایی که بر روی اعداد گویا می‌توانیم انجام دهیم عبارتند از جمع ، تفریق ، ضرب و تقسیم و غیره. میخواهیم این کلاس را پیاده‌سازی کنیم و از آن در برنامه استفاده کنیم.

عدد گویا مثل $\frac{4}{5}$ و $\frac{3}{2}$ و ...

نام کلاس : Rational

اعضای داده‌ای عبارتند از : صورت و مخرج

توابع عملیاتی عبارتند از :

- سازنده‌های مورد نیاز (بدون پارامتر و با پارامتر)
- ساده‌سازی صورت و مخرج
- تابع جمع کننده دو عدد گویا (در این تابع دو کلاس به عنوان ورودی تابع در نظر گرفته می‌شوند و نتیجه یک کلاس گویا را بر می‌گرداند)
- تابع تفریق دو عدد گویا
- تابع ضرب دو عدد گویا
- تابع تقسیم دو عدد گویا بر یکدیگر

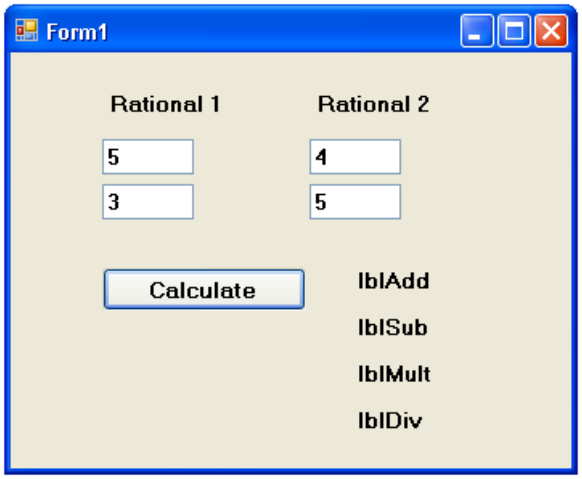
```
class Rational
{
        private int soorat, makhraj;

        public Rational() { soorat = 0; makhraj = 1; }
        public Rational(int a) { soorat = a; makhraj = 1; }
        public Rational(int a, int b) { soorat = a; makhraj = b; normal(); }
        public void set_data( int a, int b ) { soorat = a;  makhraj = b; normal(); }
        public void normal()
        {
            int max = (soorat > makhraj) ? soorat : makhraj;
            for ( int i=1; i<=max; i++ )
                if (soorat % i == 0 && makhraj % i == 0)
                {
                    soorat = soorat / i;
                    makhraj = makhraj / i;
                }
        }
```

```
    public Rational Add(Rational x, Rational y)
    {
        Rational temp = new Rational();
        temp.soorat = x.soorat * y.makhraj + y.soorat * x.makhraj;
        temp.makhraj = x.makhraj * y.makhraj;
        temp.normal();
        return temp;
    }
    public Rational Sub(Rational x, Rational y)
    {
        Rational temp = new Rational();
        temp.soorat = x.soorat * y.makhraj - y.soorat * x.makhraj;
        temp.makhraj = x.makhraj * y.makhraj;
        temp.normal();
        return temp;
    }
    public Rational Mult(Rational x, Rational y)
    {
        Rational temp = new Rational();
        temp.soorat = x.soorat * y.soorat;
        temp.makhraj = x.makhraj * y.makhraj;
        temp.normal();
        return temp;
    }
    public Rational Div(Rational x, Rational y)
    {
        Rational temp = new Rational();
        temp.soorat = x.soorat * y.makhraj;
        temp.makhraj = x.makhraj * y.soorat;
        temp.normal();
        return temp;
    }

    public int get_soorat()
    {
        return soorat;
    }

    public int get_makhraj()
    {
        return makhraj;
    }
}
```



کد فرم را به صورت زیر بنویسید :

```
private void btnCalculate_Click(object sender, EventArgs e)
{
   int x1 = Convert.ToInt32(textBox1.Text);
   int y1 = Convert.ToInt32(textBox2.Text);
   int x2 = Convert.ToInt32(textBox3.Text);
   int y2 = Convert.ToInt32(textBox4.Text);
   Rational r1 = new Rational(x1, y1);
   Rational r2 = new Rational(x2, y2);
   Rational rAdd = new Rational();
   Rational rSub = new Rational();
   Rational rMul = new Rational();
   Rational rDiv = new Rational();
   rAdd = rAdd.Add(r1, r2);
   rSub = rSub.Sub(r1, r2);
   rMul = rMul.Mult(r1, r2);
   rDiv = rDiv.Div(r1, r2);
   lblAdd.Text  = rAdd.get_soorat().ToString() + " / " + rAdd.get_makhraj().ToString();
   lblSub.Text  = rSub.get_soorat().ToString() + " / " + rSub.get_makhraj().ToString();
   lblMult.Text = rMul.get_soorat().ToString() + " / " + rMul.get_makhraj().ToString();
   lblDiv.Text  = rDiv.get_soorat().ToString() + " / " + rDiv.get_makhraj().ToString();
}
```

۱۰– تمرین در خانه

۱- کلاس اعداد مختلط را پیاده سازی کنید. اعداد مختلط از دو قسمت موهومی و حقیقی تشکیل می‌شوند .عملیاتهایی مثل جمع ،تفریق ،ضرب و تقسیم را روی این کلاس پیاده‌سازی کنید. نام کلاس را Complex بگیرید. C = (X + iY)

۲– کلاسی برای دانشجو طراحی کنید و از مطالبی که در فصل‌های قبلی یاد گرفتید ، یک مثال به طور دلخواه بنویسید.

فصل هفتم : بانک اطلاعاتی

۱- بانک‌های اطلاعاتی
۲- سیستم مدیریت بانک اطلاعاتی
۳- مفهوم بانک اطلاعاتی
۴- سیستم مدیریت بانک اطلاعاتی SQL Server 2005
۵- مشخصات فیلدهای یک جدول
۶- ایجاد بانک اطلاعاتی و تعریف جدول در SQL Server 2005
۷- معرفی دستورات SQL
۸- فضای نام sqlClient
۹- کلاس sqlConnection
۱۰-کلاس sqlDataAdaptor
۱۱-کلاس DataSet
۱۲-کلاس sqlCommand
۱۳-تمرین در خانه

۱– بانک‌های اطلاعاتی :

یکی از مهمترین بخش هر برنامه کاربردی بانک اطلاعاتی آن است که با چگونگی ذخیره و بازیابی داده‌ها سروکار دارد. در این فصل ضمن تشریح مفهوم بانک‌اطلاعاتی ، به شیوه پردازش آن در سی‌شارپ می‌پردازیم. در این فصل با موضوعات زیر آشنا خواهید شد :

- سیستم مدیریت بانک اطلاعاتی
- بانک اطلاعاتی SQL Server
- دستورالعمل‌های SQL برای تقاضا از بانک اطلاعاتی
- ارتباط بانک اطلاعاتی با سی‌شارپ

۲– سیستم مدیریت بانک اطلاعاتی :

بانک اطلاعاتی از نظر فیزیکی فایل‌هایی هستند که در سیستم‌های کامپیوتری برای ذخیره و بازیابی داده‌ها به کار می‌روند. برنامه‌های کاربردی مستقیماً نمی‌توانند این فایل‌ها را دستکاری کنند ، بلکه برای پردازش این فایل‌ها فرمان‌هایی را به سیستم مدیریت بانک اطلاعاتی (DBMS) صادر و پاسخ را دریافت می‌کنند. نمونه‌هایی از سیستم‌های مدیریت بانک اطلاعاتی ، SQL Server و Oracle و Access و غیره هستند. سیستم‌های مدیریت بانک اطلاعاتی برنامه‌نویس را از انجام کارهای دشوار و خسته‌کننده‌ای که در پردازش بانک اطلاعاتی متداول است رها می‌سازند.

۳– مفهوم بانک اطلاعاتی :

تقریباً تمام بانک‌های اطلاعاتی نوین از مدل رابطه‌ای (جدولی) استفاده می‌کنند و در نتیجه بانک اطلاعاتی رابطه‌ای نامیده می‌شوند. مهم‌ترین عنصر هر بانک اطلاعاتی رابطه‌ای **جدول** است. ستون‌های هر جدول را **فیلد** و سطرهای هر **جدول** را رکورد گویند. به عنوان مثال اگر جدولی از دانشجویان یک کلاس تشکیل دهیم ، مشخصات کامل هر دانشجو که یک سطر از جدول را اشغال می‌کند ، یک رکورد نام دارد و هر یک از مشخصات دانشجو مثل نام ، نام خانوادگی که یک ستون از جدول را اشغال می‌کند یک فیلد نامیده می‌شود.

۴– سیستم مدیریت بانک اطلاعاتی SQL Server 2005 :

برای ایجاد و دستکاری بانک اطلاعاتی از سیستم مدیریت بانک اطلاعاتی SQL Server 2005 استفاده می‌کنیم. پس از اینکه ساختار بانک اطلاعاتی را در SQL Server تعیین کردیم ، با استفاده از امکاناتی که در سی‌شارپ وجود دارد آن را پردازش می‌کنیم. به عنوان مثال جدول زیر مشخصات تعدادی از دانشجویان را نشان می‌دهد.

شماره دانشجویی	نام‌خانوادگی	جنسیت	معدل	تعداد واحد
870856577	صبح خیز	مرد	۱۹/۲۵	۸۶
890544555	طاهری	زن	۱۹/۷۵	۷۰

در این جدول مشخصات دو دانشجو آمده است که می‌گوییم این جدول دارای دو رکورد است. یک رکورد مربوط به دانشجویی به نام‌خانوادگی صبح خیز و رکورد دیگر مربوط به دانشجویی به نام‌خانوادگی طاهری است. هر رکورد دارای پنج فیلد است که ستون‌های جدول را تشکیل می‌دهند. هر یک از این فیلدها عبارتند از شماره دانشجویی ، نام‌خانوادگی ، جنسیت ، معدل و تعداد واحد.

۵– مشخصات فیلدهای یک جدول :

همانطور که دیدید فیلد بخشی از رکورد است که اطلاعات را ذخیره می‌کند. هر فیلد دارای سه مشخصه به شرح زیر است :

۱- نام فیلد ۲- نوع فیلد ۳- اندازه

برای نامگذاری فیلدها از ترکیبی از حروف الفبا و ارقام استفاده می‌شود که باید با حروف شروع شوند. مثل FirstName و LastName. نام هر فیلد حداکثر می‌تواند ۶۴ کارکتر باشد.

تذکر : در نام‌گذاری فیلدها نبایستی از کلمات کلیدی مثل Select ، Where ، Count ، Date ، Time ، Max و ... استفاده کنید.

نوع فیلد مشخص می‌کند چه نوع داده‌ای باید در فیلد ذخیره شود. در SQL Server 2005 ، ۲۹ نوع داده وجود دارد که برخی از آنها را معرفی می‌کنیم :

Text : این نوع از همه متداول‌تر است و یک نوع داده ۱۶ بایتی برای ذخیره کردن متن است و تعداد کارکترهایی که می‌تواند در خود قرار دهد به اندازه ۲ به توان ۱۶ یعنی ۶۵۵۳۶ کارکتر است.

nChar : این نوع داده برای ذخیره رشته‌ای با طول ثابت بکار می‌رود. اندازه این فیلد می‌تواند بین ۱ تا ۴۰۰۰ کارکتر باشد.

varChar : این نوع فیلد برای ذخیره رشته با طول ثابت بکار می‌رود. اندازه این فیلد می‌تواند بین ۱ تا ۸۰۰۰ کارکتر باشد.

nVarChar : این نوع فیلد برای ذخیره رشته با طول ثابت بکار می‌رود. اندازه این فیلد می‌تواند بین ۱ تا ۴۰۰۰ کارکتر باشد.

Char : این نوع فیلد نیز برای ذخیره رشته با طول ثابت بکار می‌رود و اندازه این فیلد می‌تواند بین ۱ تا ۸۰۰۰ کارکتر باشد.

Bit : این نوع داده برای ذخیره اعداد منطقی (بولین) بکار می‌رود. یک به عنوان درست و صفر به عنوان نادرست.

tinyInt : این نوع داده برای ذخیره اعداد صحیح ۱ بایتی بکار می‌رود.

smallInt : این نوع داده برای ذخیره اعداد صحیح ۲ بایتی بکار می‌رود.

Int : این نوع داده برای ذخیره اعداد صحیح ۴ بایتی بکار می‌رود.

BigInt : این نوع داده برای ذخیره اعداد صحیح ۸ بایتی بکار می‌رود.

Real : این نوع داده برای ذخیره اعداد اعشاری ۴ بایتی بکار می‌رود.

Float : این نوع داده برای ذخیره اعداد اعشاری ۸ بایتی بکار می‌رود.

Decimal : این نوع داده برای ذخیره اعداد اعشاری ۹ بایتی بکار می‌رود.

Numeric : این نوع داده برای ذخیره اعداد صحیح یا اعشاری ۹ بایتی بکار می‌رود.

SmallDateTime : این نوع داده برای ذخیره تاریخ یا زمان بکار می‌رود و ۴ بایتی است.

DateTime : این نوع داده برای ذخیره تاریخ و زمان به کار می‌رود و ۸ بایتی است.

smallMoney : این نوع داده برای ذخیره اعداد ارزی یا پولی بکار می‌رود و ۴ بایتی است.

Money : این نوع داده برای ذخیره اعداد ارزی یا پولی بکار می‌رود و ۸ بایتی است.

Image : این نوع فیلد برای ذخیره تصاویر به صورت باینری بکار می‌رود.

۶– ایجاد بانک اطلاعاتی و تعریف جدول در SQL Server 2005 :

همانطور که گفته شد , جدول یکی مهم‌ترین عناصر بانک اطلاعاتی رابطه‌ای است و مجموعه‌ای از جدول‌ها , بانک اطلاعاتی را تشکیل می‌دهند. اکنون با مثالی چگونگی ساخت جدول را در SQL Server شرح می‌دهیم. جدول زیر را در نظر بگیرید می‌خواهیم این جدول را در SQL Server ایجاد کنیم.

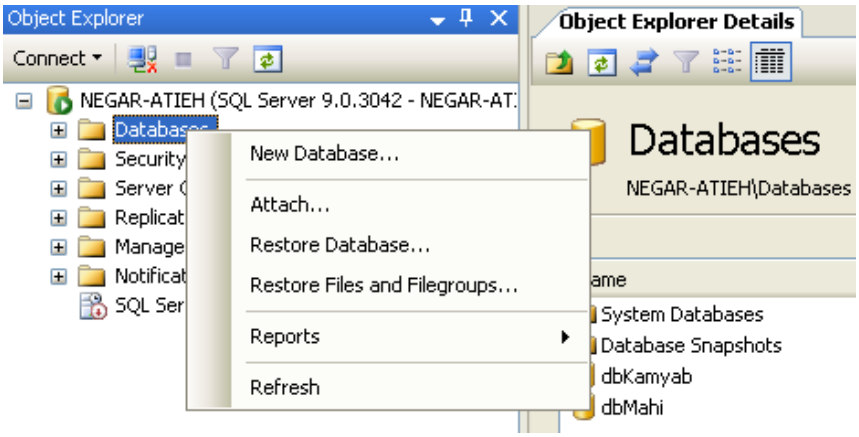
نام فیلد	شرح	نوع فیلد	طول
StNo	شماره دانشجویی	nvarChar	۱۲
Name	نام دانشجو	nvarChar	۳۵
Sex	جنسیت	bit	۱
Ave	معدل	Real	۴
NumUnit	تعداد واحد	smallInt	۲

- اگر سیستم SQL Server 2005 روی سیستم شما نصب نیست آن را نصب کنید و از مسیر زیر SQL Server Management Studio را اجرا کنید :

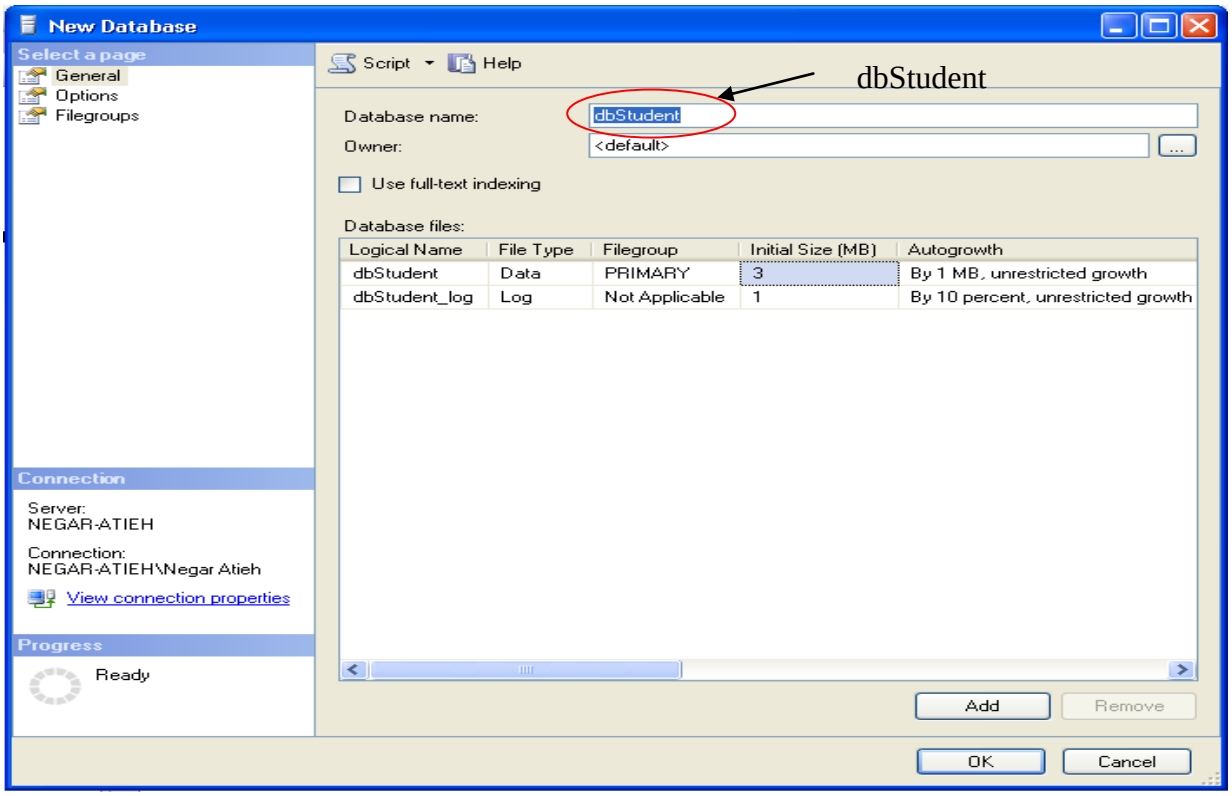
Start / All Programs / Microsoft SQL Server 2005 /

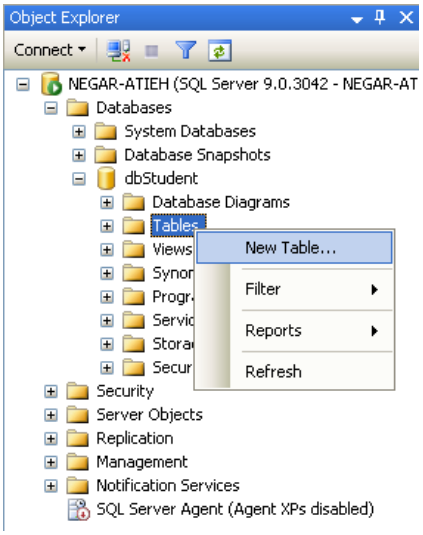
- با اجرا کردن این گزینه کادری به صورت زیر باز می‌شود : در این کادر Server Type را روی Database Engine قرار دهید و روی دکمه Connect کلیک کنید.

- در کادر زیر برای اینکه بتوانید بانک اطلاعاتی را بسازید ، رو Databases راست کلیک کنید و گزینه New Database... را انتخاب کنید :



- کادری به شکل زیر باز می‌شود : در این کادر نام بانک را وارد کنید ، مثل dbStudent. سپس روی دکمه OK کلیک کنید تا بانک ساخته شود.





- با این کار بانک dbStudent به لیست اضافه می‌شود :
- کنار dbStudent علامت + را کلیک کنید ، لیستی باز می‌شود که از این لیست ، روی Tables راست کلیک کنید و گزینه New Table را انتخاب کنید.
- با این کار پنجره جدیدی برای ایجاد جدول یا Table باز می‌شود.
- اطلاعات زیر را درون این پنجره وارد کنید :

NEGAR-ATIEH.d... dbo.Table_1* | Object Explorer Details

Column Name	Data Type	Allow Nulls
stNo	nvarchar(12)	☐
Name	nvarchar(35)	☑
Sex	bit	☑
Ave	real	☑
NumUnit	smallint	☑
		☐

- هر جدول بایستی دارای یک کلید اصلی باشد ، برای اینکه بتوانیم فیلد stNo را به عنوان کلید اصلی بگیریم ، روی این فیلد راست کلیک کنید و گزینه Set Primary Key را انتخاب کنید. حال بایستی بعد از طراحی جدول مورد نظر را ذخیره کنید : برای این کار روی آیکون ذخیره کلیک کنید و نام جدول جدید را وارد کنید.(نام جدول را tStudent بدهید و روی دکمه Ok کلیک کنید)

۷– معرفی دستورات SQL :

در ویژوال بیسیک می‌توان برای ایجاد تقاضا از بانک اطلاعاتی از دستورات زبان پرس‌وجوی SQL استفاده کرد. با استفاده از دستورات SQL می‌توان جدولی ایجاد کرد ، جدولی را ویرایش کرد و یا تقاضایی را بر روی جدولی اعمال کرد. بعضی از دستورات SQL عبارتند از :

SELECT – UPDATE – INSERT – DELETE. هرچند که دستورات SQL بیشتر از این چهار دستور است ولی ما به همین چهار دستور بسنده می‌کنیم و به توضیح همین چهار دستور می‌پردازیم :

۷–۱. دستور SELECT ... FROM :

این دستور تمام یا تعدادی از رکوردهای یک جدول را بازیابی می‌کند و به صورت زیر بکار می‌رود :

SELECT {* یا نام فیلدهای‌جدول یا جداول} FROM {نام جدول یا جداول}
WHERE شرطها
Order By نام فیلدها

اگر * در جلوی SELECT گذاشته شود تمام فیلدها بازیابی می‌شوند. اگر نام چندین فیلد ذکر شود بایستی با کاما از هم جدا شوند.

دستور Where اختیاری است و اگر بخواهیم رکوردهایی از جدول را طبق یک شرط بازیابی کنیم از آن استفاده می‌کنیم اگر چندین شرط در Where موجود باشند با استفاده از عملگرهای منطقی And و یا Or بایستی آنها را ترکیب کنیم.

دستور Order By برای مرتب کردن نتیجه بازیابی بر اساس یک یا چند فیلد بکار می‌رود.

برای مثال دستور زیر را در نظر بگیرید :

```
SELECT StNo, Name, Ave  FROM STUDENT
WHERE Ave > 15
ORDER BY StNo
```

کوئری بالا شماره دانشجویی و نام و معدل دانشجویانی را از جدول Student بازیابی می‌کند که معدل آنها از ۱۵ بیشتر است و بر اساس مقادیر فیلد StNo آنها را مرتب می‌کند.

در کوئری بالا اگر به جای خط اول می‌نوشتیم : SELECT * FROM STUDENT تمامی فیلدهای جدول بازیابی می‌شد.

۷-۲. دستور INSERT :

این دستور یک یا چند رکورد را به جدول اضافه می‌کند. این دستور به صورت زیر بکار می‌رود :

INSERT INTO نام‌جدول (نام‌فیلد۱ , نام‌فیلد۲ , ... , نام‌فیلدn)

VALUES (مقدار۱ , مقدار۲ , ... مقدار n)

برای مثال دستور زیر یک رکورد با مشخصات جدید به جدول STUDENT اضافه می‌کند :

```
INSERT INTO STUDENT ( StNo , Name , Sex , Ave , NumUnit )
Values ( '880556555' , 'Ahmad' , '1' , '18.45' , '78' )
```

۷-۳. دستور UPDATE :

این دستور برای اصلاح مقادیر رکورد یا رکوردهایی از جدول به کار می‌رود و به صورت زیر بکار می‌رود :

UPDATE نام‌جدول

SET Field1 = Value1 , Field2 = Value2 , … Fieldn = Valuen

WHERE شرط‌ها

برای مثال دستور زیر معدل دانشجوی با شماره دانشجویی ۸۷۰۰۸ را به ۱۳/۷۸ تغییر می‌دهد :

```
UPDATE STUDENT
SET  Ave = '13.78'
WHERE StNo = '880556555'
```

۷-۴. دستور DELETE :

این دستور تقاضایی را برای حذف رکورد یا رکوردهایی از جدول ایجاد می‌کند و به صورت زیر استفاده می‌شود :

DELETE FROM نام‌جدول

WHERE شرط‌ها

برای مثال دستور زیر تمامی دانشجویان دختر را از جدول حذف می‌کند :

```
DELETE FROM STUDENT
WHERE Sex = '0'
```

۸- فضای نام sqlClient : در زبان سی‌شارپ برای اتصال به بانک اطلاعاتی از تکنیک ADO.NET استفاده می‌شود. که برای اتصال به بانک اطلاعاتی SQL Server بایستی از فضای نام sqlClient و برای اتصال به بانک‌های اطلاعاتی غیر از SQL Server بایستی از OleDb استفاده کنیم. ما بایستی این فضای نام‌ها را برای استفاده از کلاس‌های دسترسی به بانک اطلاعاتی به صورت زیر به برنامه اضافه کنیم :

```
using System.Data;
using System.Data.SqlClient;
```

۹- کلاس sqlConnection : از این کلاس برای اتصال به بانک اطلاعاتی SQL Server استفاده می‌شود. طرز استفاده آن در زیر آمده است :

```
SqlConnection con1 = new SqlConnection("Integrated Security=TRUE; Initial Catalog= نام‌بانک ");
con1.Open();
دستورات//
con1.Close();
```

روش دیگری نیز برای اتصال وجود دارد که بایستی ابتدا نام کاربری و رمز عبوری در SQL Server تعریف کنیم و کد زیر را بنویسیم :

```
SqlConnection con1 = new SqlConnection("SERVER=سرور; Database=نام‌بانک; UID=نام‌کاربر; PWD=رمز");
con1.Open();
دستورات//
con1.Close();
```

۱۰- کلاس sqlDataAdaptor :

از این کلاس برای اجرای دستورات SQL و بازیابی جداول بانک اطلاعاتی استفاده می‌شود. این کلاس یک واسطه میان کلاس DataSet و بانک اطلاعاتی می‌باشد. طرز استفاده از این کلاس به صورت زیر است :

```
SqlConnection con1 = new SqlConnection("Integrated Security=TRUE; Initial Catalog= dbStudent");
con1.Open();
SqlDataAdapter da1 = new SqlDataAdapter("SELECT * FROM tStudent",con1);
con1.Close();
```

۱۱– کلاس DataSet : از این کلاس توسط sqlDataAdaptor می‌توانیم جدول یا رکوردهایی که بازیابی کرده‌ایم درون کنترل‌های سی‌شارپ مثل DataGridView ، listBox ، comboBox و غیره نمایش دهیم.

```
SqlConnection con1 = new SqlConnection("Integrated Security=TRUE; Initial Catalog= dbStudent");
con1.Open();
SqlDataAdapter da1 = new SqlDataAdapter("SELECT * FROM tStudent",con1);
DataSet ds1 = new DataSet();
da1.Fill(ds1, "TEMP");
dataGridView1.DataSource = ds1;
dataGridView1.DataMember = "TEMP";
con1.Close();
```

در کد بالا از متد Fill برای پر کردن ds1 که شیء از کلاس DataSet می‌باشد استفاده شده است. ما درون ds1 جدولی که توسط dataAdaptor بازیابی شده است را با نام مجازی TEMP ذخیره کردیم. حال با استفاده از ds1 و جدول مجازی TEMP می‌توانیم رکوردها را درون کنترل‌های سی‌شارپ نمایش دهیم.

برای اینکه DataGridView به DataSet متصل شود بایستی خاصیت DataSource را به ds1 مقداردهی کنیم. و خاصیت DataMember آن را برابر نام جدول مجازی ذخیره شده در ds1 که همان TEMP می‌باشد قرار دهیم.

برای اینکه کنترل‌های دیگری مثل listBox ، comboBox و غیره را به جدول بازیابی اتصال دهیم بایستی DataSource آن را برابر ds1 و خاصیت DisplayMember آن را به فیلد جدول مجازی مقدار دهی کنیم به صورت زیر :

```
listBox1.DataSource = ds1;
listBox1.DisplayMember = "TEMP.Average";
```

۱۲– کلاس sqlCommand : از این دستور برای اجرای دستورات SQL از قبیل INSERT ، UPDATE ، DELETE و غیره روی بانک اطلاعاتی استفاده می‌شود. طرز استفاده از این دستور را با مثال زیر ببینید :

```
SqlConnection con1 = new SqlConnection("Integrated Security=TRUE; Initial Catalog= dbStudent");
con1.Open();
String strQuery = "INSERT INTO tStudent VALUES( '880987115','Iman',1,15.25,86 )";
SqlCommand com1 = new SqlCommand(strQuery, con1);
com1.ExecuteNonQuery();
con1.Close();
```

مثال ۱: برنامه‌ای می‌نویسیم که بیشتر مطالبی که تاکنون در مورد بانک اطلاعاتی یاد گرفتیم را دارا باشد.

بانک اطلاعاتی ما همان DBCollege و دارای یک جدول Student می‌باشد. فیلدهای آن عبارت بودن از : stNo ، Name ، Sex ، Average و numUnits می‌باشد.

برنامه ما قابلیت انجام Insert ، Delete ، Search را دارا می‌باشد. همچنین لیستی از دانشجویان را نیز نمایش می‌دهد.


```
private void load_database()
{
  SqlConnection con1 = new SqlConnection("Integrated Security=TRUE; Initial Catalog= dbStudent");
  con1.Open();
  SqlDataAdapter da1 = new SqlDataAdapter("SELECT * FROM tStudent", con1);
  DataSet ds1 = new DataSet();
  da1.Fill(ds1, "TEMP");
  dataGridView1.DataSource = ds1;
  dataGridView1.DataMember = "TEMP";
```

```
  con1.Close();
}
```

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

```
private void Search_By_No_Click(object sender, EventArgs e)
{
  SqlConnection con1 = new SqlConnection("Integrated Security=TRUE; Initial Catalog= dbStudent");
  con1.Open();
  SqlDataAdapter da1 = new SqlDataAdapter("SELECT * FROM tStudent WHERE stNo = '"
                                          + textBox1.Text + "'", con1);
  DataSet ds1 = new DataSet();
  da1.Fill(ds1, "TEMP");
  if (this.BindingContext[ds1, "TEMP"].Count > 0)
  {
     dataGridView1.DataSource = ds1;
     dataGridView1.DataMember = "TEMP";
  }
  else
  {
     MessageBox.Show("دانشجویی با این شماره یافت نشد");
  }
  con1.Close();
}
```

```
private void Delete_By_No_Click(object sender, EventArgs e)
{
  SqlConnection con1 = new SqlConnection("Integrated Security=TRUE; Initial Catalog= dbStudent");
  con1.Open();
  String strQuery = "DELETE FROM tStudent WHERE stNo = '" + textBox1.Text + "'";
  SqlCommand com1 = new SqlCommand(strQuery,con1);
  com1.ExecuteNonQuery();
  con1.Close();
  load_database();
}
```

```
private void btnINSERT_Click(object sender, EventArgs e)
{
  SqlConnection con1 = new SqlConnection("Integrated Security=TRUE; Initial Catalog= dbStudent");
  con1.Open();
  String strQuery = String.Format("INSERT INTO tStudent VALUES ('{0}','{1}',{2},{3},{4})",
                                  textBox2.Text,textBox3.Text,
                                  Convert.ToByte(checkBox1.Checked),
                                  textBox4.Text,
                                  textBox5.Text);
  SqlCommand com1 = new SqlCommand(strQuery, con1);
  com1.ExecuteNonQuery();
  con1.Close();
  load_database();
}
```

مثال۲ : در برنامه قبلی کد دکمه Insert را می‌خواهیم طوری تغییر دهیم که قبل از ثبت اطلاعات بررسی کند که شماره دانشجویی تکراری نباشد ، اگر تکراری نبود رکورد دانشجوی جدید ذخیره شود. در غیر اینصورت با یک پیغام تکراری بودن دانشجو را نمایش دهد.

```
private void btnINSERT_Click(object sender, EventArgs e)
{
  SqlConnection con1 = new SqlConnection("Integrated Security=TRUE; Initial Catalog=dbStudent");
  con1.Open();

  SqlDataAdapter da1 = new SqlDataAdapter("SELECT * FROM tStudent WHERE stNo = '" +
                                          textBox2.Text  + "'", con1);
  DataSet ds1 = new DataSet();
  da1.Fill(ds1, "TEMP");
  if ( ds1.Tables["TEMP"].Rows.Count == 0 )
  {
     String strQuery = String.Format("INSERT INTO tStudent VALUES ('{0}','{1}',{2},{3},{4})",
                                  textBox2.Text, textBox3.Text,
                                  Convert.ToByte(checkBox1.Checked), textBox4.
```

```
                                        Text, textBox5.Text);
      SqlCommand com1 = new SqlCommand(strQuery, con1);
      com1.ExecuteNonQuery();
    }
    else
    {
      MessageBox.Show("دانشجویی با این شماره از قبل ثبت نام کرده است");
    }
    con1.Close();
    load_database();
  }
```

مثال۳ : دکمه ای به فرم اضافه کنید که معدل کل دانشجویان را بدست بیاورد و با پیغامی نمایش دهد.

```
  private void btnCalcAverage_Click(object sender, EventArgs e)
  {
     qlConnection con1 = new SqlConnection("Integrated Security=TRUE; Initial Catalog=dbStudent");
     con1.Open();
     String strQuery = String.Format("SELECT AVG(Average) FROM tStudent");
     SqlCommand com1 = new SqlCommand(strQuery, con1);
     String str = com1.ExecuteScalar().ToString();
     MessageBox.Show(str);
     con1.Close();
     load_database();
  }
```

مثال۴ : تابع **Load_Database** را طوری تغییر دهید که هنگام نمایش جدول ، عنوان فیلدها را فارسی و عرض ستونها را نیز به اندازه مورد نیاز تغییر دهد.

```
  private void load_database()
  {
    SqlConnection con1 = new SqlConnection("Integrated Security=TRUE; Initial Catalog=dbStudent");
    con1.Open();
    SqlDataAdapter da1 = new SqlDataAdapter("SELECT * FROM tStudent", con1);
    DataSet ds1 = new DataSet();
    da1.Fill(ds1, "TEMP");
    dataGridView1.DataSource = ds1;
    dataGridView1.DataMember = "TEMP";
    dataGridView1.Columns[0].HeaderText = "کد دانشجویی";
    dataGridView1.Columns[1].HeaderText = "نام";
    dataGridView1.Columns[2].HeaderText = "جنسیت";
    dataGridView1.Columns[3].HeaderText = "معدل";
    dataGridView1.Columns[4].HeaderText = "تعداد واحد";
    dataGridView1.Columns[0].Width = 100;
    dataGridView1.Columns[1].Width = 120;
    dataGridView1.Columns[2].Width = 50;
    dataGridView1.Columns[3].Width = 50;
    dataGridView1.Columns[4].Width = 75;
    con1.Close();
  }
```

مثال۵ : می‌خواهیم بانک اطلاعاتی به نام db_Forooshgah با یک جدول t_Kala با فیلدهای زیر بسازیم :

نام فیلد	شرح	نوع فیلد	طول
Kala_no	شماره کالا	nvarChar	20
Kala	نام کالا	nvarChar	30
Number	تعداد کالا	smallInt	2
Price	قیمت کالا	numeric	9

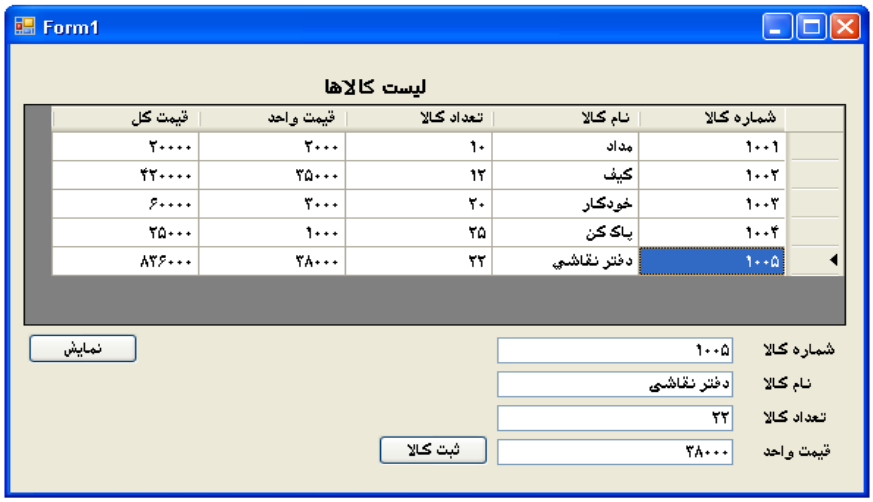
در این برنامه می‌خواهیم چهار جعبه متن و یک دکمه برای ثبت اطلاعات کالای جدید به فرم اضافه کنیم.

برنامه قابلیت این را داشته باشد که برای نمایش اطلاعات در دیتاگرید ، ستونی به جدول اضافه کند که از ضرب تعداد کالا در قیمت کالا ستونی با نام مبلغ کل بدست آورد

```
private void btn_Display_Click(object sender, EventArgs e)
{
  SqlConnection con1 = new SqlConnection("Integrated Security=TRUE; Initial Catalog=DB_Forooshgah");
  con1.Open();
  String strQuery = "SELECT Kala_no As [کالا شماره], Kala As [کالا نام], " +
                    "Number As [کالا تعداد], Price As [واحد قیمت], " +
                    "(Number*Price) As [کل قیمت] FROM T_Kala";
  SqlDataAdapter da1 = new SqlDataAdapter(strQuery, con1);
```

```
  DataSet ds1 = new DataSet();
  da1.Fill(ds1, "TEMP");
  dataGridView1.DataSource = ds1;
  dataGridView1.DataMember = "TEMP";
  con1.Close();
}
```

```
private void btn_SABT_Click(object sender, EventArgs e)
{
  SqlConnection con1 = new SqlConnection("Integrated Security=TRUE; Initial Catalog=DB_Forooshgah");
  con1.Open();
  SqlDataAdapter da1 = new SqlDataAdapter("SELECT * FROM T_Kala WHERE Kala_no = '" +
                                          textBox1.Text + "'", con1);
  DataSet ds1 = new DataSet();
  da1.Fill(ds1, "TEMP");
  if (ds1.Tables["TEMP"].Rows.Count == 0)
  {
     String strQuery = String.Format("INSERT INTO T_Kala VALUES ('{0}','{1}',{2},{3})",
                                     textBox1.Text, textBox2.Text, Convert.ToInt32(textBox3.Text),
                                     Convert.ToInt64(textBox4.Text));
     SqlCommand com1 = new SqlCommand(strQuery, con1);
     com1.ExecuteNonQuery();
  }
  else
  {
     MessageBox.Show("کالایی با این شماره از قبل وجود دارد");
  }
  con1.Close();
  btn_Display_Click(null, null);
}
```



تمرین در خانه :

۱– در مثال ۵ دکمه‌ای به فرم اضافه کنید که بتواند جمع کل قیمت‌ها را در برچسبی نمایش دهد.

۲– برنامه‌ای بنویسید که از دو جدول Student و Sabtenam استفاده کند. فیلد مشترک دو جدول را شماره دانشجویی بگیرید. این برنامه قابلیت درج رکورد دانشجوی جدید ، ثبت نام دانشجوی جدید را داشته باشد ، نمایش اطلاعات ثبت نام دانشجو به طوری که مشخصات دانشجو در آن نیز نمایش داده شود.(از منوها استفاده کنید).