

متغیر چیست؟متغیر مکانی از حافظه است که مقداری را درون خود با توجه به نوع داده نگهداری میکند و دارای یک نام است.

انواع داده ها عبارتند از :

انواع داده		
یک بایت	عدد صحیح	byte:
دو بایت	عدد صحیح	short:
چهار بایت	عدد صحیح	int :
هشت بایت	عدد صحیح	long:
دقت تا هفت رقم بعد از اعشار	عدد اعشاری	float:
دقت تا پانزده رقم بعد از اعشار	عدد اعشاری	double:
دقت تا ۲۸ رقم بعد از اعشار	عدد اعشاری	decimal:
true , false	منطقی	bool:
یک کاراکتر	کاراکتر	char:
هرنوع داده	هر نوع داده	object:
نوع داده رشته ای	رشته ای	string:

Char: برای نگهداری یک کاراکتر است و کاراکتر داخل تک کوتیشن قرار میگیرد.

String: نوع داده رشته ای است که شامل توالی کاراکتر های یونیکد است و داخل دابل کوتیشن قرار میگیرد.

قوانین نام گذاری متغیر ها:

۱. برای نام گذاری متغیر ها میتوان از حروف الفبا و اعداد و آندرلاین استفاده کرد.

۲. نباید نام متغیر با عدد شروع شود.

۳. نباید از کلمات رزرو شده و کلمات کلیدی سیستم استفاده شود.

۴. تنها کاراکتر مجاز آندرلاین است.

➤ نکته: زبان برنامه نویسی سی شارپ به حروف بزرگ و کوچک حساس است.

هنگام تايپ کاربر برنامه نويس از دکمه های زیر استفاده می کند.

- ENTER
- SPACE
- TAB

➤ تعريف يك متغير

مقدار متغير = نام متغير نوع متغير

مثال

`bool a=true;` `double y=۱۲,۷۵;` `int x=۵;`

✓ نکته: در انتهای تعريف متغير بايد از سمی کالن استفاده کرد.

➤ ثابت ها و تعريف آن ها:

ثابت ها مانند متغير ها هستند مکانی از حافظه برای نگهداری مقادير می باشند با اين تفاوت که مقدار آن ها در طول اجرای برنامه تغيير نمی کند.

مقدار = نام ثابت نوع داده **Const**

مثال `Const int x=۱۲;`

روش های تبديل داده :

۱_ تبديل داده با تابع **parse**. برای تبديل رشته ها به اعداد به کار می رود :

`Int A = int.parse(textBox1.Text);`

۲_ تبديل داده با استفاده از توابع **convert** : برای تبديل همه ی داده ها به يکديگر به کار می رود :

`int a = Convert.ToInt32(textBox1.Text);`

نکته : `int ۱۶` (برای اعداد کوچک) ، `int ۳۲` (برای اعداد متوسط) و `int ۶۴` (برای اعداد بزرگ) به کار می روند.

۳_ تبديل داده با **Type cast** : برای تبديل خانواده های همونوع به يکديگر به کار می رود :

`Double M=۱۲/۵;`

`Int n = (int) M;`

مقداری که الان داخل متغیر n قرار میگیرد برابر ۱۲ است زیرا رقم اعشار آن حذف میشود و قسمت صحیح عدد داخل متغیر قرار میگیرد.

نکته: type cast برای تبدیل داده هایی که از یک خانواده هستند (مثلا خانواده اعداد ریاضی) به کار برده میشود. در مثال بالا متغیر M از نوع double می باشد. در تبدیل type cast ما به سیستم اعلام می کنیم که M را به چشم integer نگاه کن و مقدار آنرا پس از تبدیل داخل n قرار بده.

باید به این نکته توجه داشت که رنج عدد اعشاری که می خواهیم به int تبدیل کنیم نباید از عدد integer بیشتر شود. مثلاً عدد اعشاری برای تبدیل به int باید در رنج ۴ بایت باشد.

➤ آرایه ها

یک مکانی از حافظه است که بستگی دارد به نوع داده ای که تعریف می کنیم مجموعه ای از مقادیر هم نوع خود را نگهداری میکند.

➤ تعریف آرایه

; [تعداد عناصر] نوع آرایه = new نام آرایه [نوع آرایه]

مثال: `Int [] A = new int [۷];`

در مثال بالا یک آرایه با ۷ خانه و اندیس صفر تا شش ساخته شد. دقت فرمایید اندیس آرایه از صفر شروع می شود.

➤ مقدار دهی خانه های آرایه

; مقدار=[شماره اندیس] نام آرایه

مثال `A[۲]=۱۰;`

شماره اندیس	۰	۱	۲	۳	۴	۵	۶
شماره خانه	اول	دوم	سوم	چهارم	پنجم	ششم	هفتم
مقدار			۱۰				

آرایه ی A

➤ آرایه ی دو بعدی

; [ستون و سطر] نوع آرایه = new نام آرایه [,] نوع آرایه

مثال `Int [,] x=new int [۲,۳];`

آرایه ی دو بعدی با دو سطر و سه ستون.

➤ مقدار دهی خانه های آرایه ی دو بعدی

مقدار=[ستون و سطر]نام آرایه

مثال

X [۰,۲]=۱۵;

		۱۵

➤ **Enum نوع داده شمارشی:** برای مجموعه ای از داده ها مثل فصل ها و ماه ها و روز های هفته که ما

خودمان می خواهیم تعریفشان کنیم از نوع داده شمارشی استفاده می کنیم.

➤ **تعریف نوع داده شمارشی**

Enum نام متغیر

```
{
    مقادیر
}
```

مثال: نوع داده شمارشی روز های هفته

Enum day

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

❖ نکته: اندیس اولین عنصر نوع داده شمارشی به طور پیش فرض از صفر شروع می شود ولی

خودمان هم میتوانیم اندیس آن را تغییر دهیم.

برای اینکه اندیس یک عنصری را تغییر دهیم به صورت زیر عمل میکنیم.

شماره اندیس مورد نظر = نام عنصر

مثال : Saturday=۳;

اندیس این عنصر ۳ خواهد بود و عناصر بعد از این عنصر نیز اندیس شان تغییر خواهد کرد .

➤ **مثالی برای تعریف و مقداری دهی نوع داده شمارشی**

Day d; d=day.Friday;

Enum fasl

مثال ۲: برای نوع داده شمارشی فصل ها

```
{  
  
Bahar,tabestan,paheiz,zemestan  
  
}  
  
Fasl f;  
  
f=fasl.bahar;
```

نکته : enum را در زیر تابع سازنده تعريف می‌کنيم .

d=(day)^۳;

مقداری که در day enum شماره اندیش ۳ است را درون متغیر d قرار می‌دهيم. یعنی در این قطعه کد روز Saturday در متغیر d قرار می‌گیرد.

جلسه دوم

انواع عملگرها :

• عملگرهای محاسباتی :

_ جمع (+)

_ تفریق (-)

_ ضرب (*)

_ تقسیم (/) برای محاسبه خارج قسمت(صحیح)

_ باقی مانده تقسیم (%)

به عنوان مثال در تقسیم عدد ۱۳ بر ۳ ، عدد ۴ خارج قسمت و عدد ۱ باقی مانده تقسیم است.

مثال :

Double x=۱۳;

Double y=۲/۵;

Double t=x%y;

در مثال بالا عدد ۰/۵ در متغیر t قرار می‌گیرد.

• الحاق رشته ها :

```
String A="Ali"+"۱۲";
```

در مثال بالا رشته ۱۲ Ali در متغیر A قرار میگیرد.

• عملگر انتساب :

```
int A;
```

```
A=۱۲;
```

در مثال بالا مقدار ۱۲ در متغیر A قرار میگیرد.

• عملگرهای افزایشی و کاهششی :

° پیشوندی : ++x و --x

° پسوندی : x++ و x--

در عملگر پیشوندی ابتدا یک واحد به x اضافه شده و سپس عملیات انجام می شود.

در عملگر پسوندی ابتدا عملیات انجام شده و در آخر به مقدار x یک واحد اضافه می شود.

مثال عملگر پیشوندی:

```
int x=۵;
```

```
int y= ++x * ۲;
```

در این قطعه کد ابتدا مقدار متغیر x برابر ۶ می شود و سپس در عدد ۲ ضرب میشود و عدد ۱۲ در متغیر y قرار میگیرد.

مثال عملگر پسوندی :

```
int x=۵;
```

```
int y= x++ * ۲;
```

در این قطعه کد متغیر x با همان مقدار اولیه ۵ در عملیات محاسباتی شرکت کرده و عدد ۵ در ۲ ضرب می شود و مقدار ۱۰ در متغیر y قرار می گیرد. پس از اجرای عملیات محاسباتی مقدار متغیر x یک واحد افزایش پیدا می کند و برابر ۶ می شود.

• عملگرهای جایگزین محاسباتی :

° عملگر += : به عنوان مثال $x += y$ جایگزین قطعه کد $x = x+y$ است.

° عملگر -= : به عنوان مثال $x -= y$ جایگزین قطعه کد $x = x-y$ است.

- عملگر $/=$: به عنوان مثال $x /= y$ جايزين قطعه كد $x = x/y$ است.
- عملگر $*=$: به عنوان مثال $x *= y$ جايزين قطعه كد $x = x*y$ است.
- عملگر $\%=$: به عنوان مثال $x \% = y$ جايزين قطعه كد $x = x \% y$ است.

مثال :

`int x=۰ , y=۳;`

`int t= x+=y;`

در مثال بالا مقدار متغير x يعني عدد ۵ و مقدار متغير y يعني عدد ۳ با يكدیگر جمع شده و در نهايت مقدار ۸ در متغير t قرار ميگيرد.

• عملگرهای رابطه ای :

◦ مساوی $==$

◦ نامساوی $!=$

◦ بزرگتر $>$

◦ کوچکتر $<$

◦ بزرگتر مساوی $>=$

◦ کوچکتر مساوی $<=$

مثال: `int x=۰ , y=۳; bool t=(x==y);`

در مثال بالا ابتدا بررسی می شود كه مقدار متغير x يعني عدد ۵ برابر مقدار متغير y يعني عدد ۳ هست يا نه؟ چون عدد ۵ برابر عدد ۳ نيست پس بنا بر اين مقدار $false$ در متغير t قرار ميگيرد.

• عملگرهای منطقی :

• و (and) $\&\&$

• يا (or) $\|\|$

• نقيض (not) $!$

عملگر And :

`true && true = true`

`false && true = false`

`true && false = false`

`false && false = false`

عملگر Or :

true || true = true

false || true = true

true || false = true

false || false = false

عملگر Not :

true = false

false = true

مثال ۱ :

bool x= true , y= false;

bool t=(x && y);

با توجه به تعاریف قبلی نتیجه کد true && false میشود false بنابراین مقدار false درون متغیر t قرار میگیرد.

مثال ۲ :

int x=۳;

if (x=۴)

X++;

نکته : در شرط ها ، جواب شرط باید یک جواب منطقی true یا false باشد.

در قطعه کد بالا با خطا مواجه می شویم زیرا جمله ی شرطی ما اشتباه است و صرفاً یک دستور انتساب است. دستور if به داده ای از نوع Boolean یعنی true یا false احتیاج دارد. بنابراین برنامه با خطای زمان اجرا (کامپایل) مواجه می شود.

مثال ۳ :

int x=۳;

bool k=false;

if (k=true)

x++;

در قطعه کد بالا چون خود متغیر k یک متغیر از نوع منطقی است ، با اجرای دستور شرط ، به جای مقدار اولیه ی خودش یعنی false ، مقدار true داخل k ریخته می شود و چون شرط برقرار است (جواب جمله ی شرط true است) مقدار متغیر x یک واحد افزایش پیدا می کند و می شود ۴ .

ولی اگر شرط برقرار نبود و جواب جمله ی شرط برابر false میشد ، مقدار متغیر x همان عدد ۳ باقی می ماند ، چون برنامه از روی دستورات مربوط به if می پرید.

مثال ۴ : `int k=۳;`

`bool x=true;`

`bool y=false;`

`if (x && (y=true))`

`k++;`

در قطعه کد بالا مقدار y برابر true میشود و مقدار x هم که true است.

طبق تعاریف قبلی نتیجه `true && true` برابر true می شود.

چون جواب جمله ی شرط true میشود پس شرط برقرار است و دستورات مربوط به if اجرا می شوند و مقدار متغیر k یک واحد افزایش پیدا می کند و میشود ۴ .

• عملگر شرطی :

° شکل کلی دستور:

(شرط غلط : شرط صحیح ? عبارت)

مثال: `int a=۲;`

`int t = (a==۳ ? ۴ : ۵);`

ابتدا بررسی می شود که مقدار متغیر a برابر ۳ هست یا خیر ؟ چون مقدار متغیر a برابر ۳ نیست و شرط صحیح نیست پس مقدار ۵ درون متغیر t قرار می گیرد.

اولویت عملگرها :

۱_ پرانتز ()

۲_ باقیمانده % تقسیم / ضرب *

۳_ جمع + تفریق -

۴_ بزرگتر > کوچکتر < بزرگتر مساوی >= کوچکتر مساوی <=

۵_ عملگر == و عملگر !=

۶_ عملگر && و عملگر ||

۷_ عملگر شرطی (: ?)

۸_ عملگرهای += , -= , /= , %= , *

۹_ عملگر انتساب =

نکته : اولویت عملگرهای شماره ۱ تا ۷ از چپ به راست و اولویت عملگرهای شماره ۸ و ۹ از راست به چپ می باشد.

مثال ۱ :

$$2+3*4=14$$

مثال ۲ :

$$(2*3)+(4*6/3)*(3-1)=22$$

مثال ۳ :

$$17(((4/2*6)-5)*3+2)=$$

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

namespace WindowsFormsApplication3
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        enum fasl
        {
            bahar = 14, tabestan, paeiz, zemestan
        }

        private void btnShow_Click(object sender, EventArgs e)
        {
            int a = 7;

            char a='M';
            string a = "Ali";
            int a = Convert.ToInt32(textBox1.Text);
            MessageBox.Show(a.ToString());
            -----
            int[] a = new int[3] { 14, 23, 8 };
            for (int i = 0; i < 3; i++)
                //MessageBox.Show(a[i].ToString());
        }
    }
}
```

```
listBox1.Items.Add(a[i].ToString());
```

```
-----  
int[,] a = new int[2,3] {{ 17,18,20},{80,90,100} };  
for (int i = 0; i < 2; i++)  
    for (int k = 0; k < 3; k++)  
        //MessageBox.Show(a[i].ToString());  
        listBox1.Items.Add(a[i,k].ToString());  
-----
```

```
int[] a = new int[3];  
a[0] = Convert.ToInt32(textBox1.Text);  
a[1] = 11;  
a[2] = 60;  
for (int i = 0; i < 3; i++)  
    //MessageBox.Show(a[i].ToString());  
    listBox1.Items.Add(a[i].ToString());  
-----
```

```
fasl f;  
f = fasl.paeiz;  
MessageBox.Show(f.ToString());  
fasl f;  
f = (fasl)17;  
//double a = 12.75;  
//int h = (int)a;  
MessageBox.Show(f.ToString());  
-----
```

```
int a = 2;  
int t=(a==3?7:19);  
MessageBox.Show(t.ToString());  
-----
```

```
int a = 2;  
int t=5;  
if (a == 2)  
{  
    a++;  
    MessageBox.Show(a.ToString());  
    a++;  
    MessageBox.Show(a.ToString());  
}
```

```
MessageBox.Show("Error");
```

```
t = 45;  
MessageBox.Show(t.ToString());
```

```
private void button1_Click(object sender, EventArgs e)  
{  
    string name = txtnumber1.Text;  
    string family = txtnumber2 . Text;  
    MessageBox.Show("hello " + name + " " + family);  
}-----
```

```
int a, b, c, sum, avg;  
a = Convert.ToInt32(txtnumber1.Text);  
b = Convert.ToInt32(txtnumber2.Text);  
c = Convert.ToInt32(txtnumber3.Text);  
sum = a + b + c;  
avg = sum / 3;  
MessageBox.Show(sum.ToString());  
MessageBox.Show(avg.ToString());
```

جلسه ی سوم

ساختار های کنترلی

- ساختار های تصمیم گیری (if , switch)
- ساختار های تکرار (for , foreach , while, do_while)

ساختار if :

- تک انتخابی If
- دو انتخابی If
- چند انتخابی (تودرتو) If

If تک انتخابی مثال:

```
int a=۷;
```

```
If (a= =۷)
```

```
MessageBox.show(a.ToString);
```

If دو انتخابی مثال:

```
Int a=convert.ToInt32(txtnumber.text);
```

```
If ( a%۲= =۰)
```

```
Message Box.show(“zgj”);
```

```
Else
```

```
Message Box.show(“fard”);
```

مثال: برنامه ای بنویسید که یک عدد را دریافت کرده و مشخص کند که آیا آن عدد بر ۳ و ۵ بخش پذیر است یا نه با نمایش پیغام مناسب.

```
Int num=convert.ToInt32(txtnum.text);  
If( num%۳ == ۰)  
MessageBox.show("بر ۳ بخش پذیر است");  
Else  
MessageBox.show("بر ۳ بخش پذیر نیست");  
If(num%۵==۰)  
MessageBox.show("بر ۵ بخش پذیر است");  
Else  
MessageBox.show("بر ۵ بخش پذیر نیست");
```

If چند انتخابی (تودرتو) مثال:

```
Int a=convert.ToInt32(txtnumber.text);  
If (a==۱)  
MessageBox.show("one");  
Else if (a==۲)  
MessageBox.show("two");  
Else if (a==۳)  
MessageBox.show("three");  
else  
MessageBox.show("error!");
```

چند نکته ی مهم درمورد دستورات if

۱. نباید از **در آخر شرط استفاده کنیم زیرا اگر بگذاریم شرط را تهی می بیند و می آید دستورات IF را اجرا می کند.**
۲. اگر در شرط بیش از یک دستور بعد از شرط داشتیم باید **{** استفاده کنیم.

ساختار switch:

```
Int a =convert.ToInt۳۲(txtnumber.text);
```

```
Switch (a)
```

```
{
```

```
Case ۱:
```

```
MessageBox.show(“one”);
```

```
Break;
```

```
Case ۲:
```

```
Message Box.show(“two”);
```

```
Break;
```

```
Case ۳:
```

```
Message Box.show(“three”);
```

```
Break;
```

```
Default:
```

```
Message Box.show(“error!”);
```

```
Break;
```

```
}
```

نکات مهم در دستور switch

۱. اگر در switch عبارتی قرار دادیم باید به گونه ای باشد که بتوان آن را ارزیابی کرد.
۲. مقادیر case ها نباید تکراری باشد.
۳. لازم است در انتهای هر case از دستور break استفاده شود بدون این دستور، اجرای برنامه پس از اینکه بلوک دستور العمل مربوط به آن case را اجرا کرد از ساختار switch خارج نمی شود (حلقه ی بی نهایت).
۴. دستور switch فقط مساوی بودن مقادیر را با متغیر یا عبارت بررسی می کند.

ساختار تکرار

۱. While
۲. Do while
۳. For
۴. Foreach

چه زمانی از ساختار تکرار استفاده میکنیم:

چنانچه بخواهیم دستوراتی را بیش از یکبار در برنامه تکرار کنیم از ساختار های تکرار استفاده میکنیم. **For** تعداد چرخش حلقه مشخص است. **While** تا زمانی که شرط برقرار باشد با برقرار بودن شرط وارد حلقه می شود (تعداد چرخش حلقه مشخص نیست). **Foreach** بیشتر برای مجموعه ها (آرایه ها) استفاده می شود.

: ساختار while

(شرط) While

```
{  
}
```

مثال: برنامه ی بنویسید که یک عدد از ورودی دریافت کرده مجموع رقم های آن را محاسبه و چاپ نماید.

```
Int a=convert.ToInt32(txtnumber.text);
```

```
Int sum=0;
```

```
Int R=0;
```

```
While (a>0)
```

```
{
```

```
R= a % ۱۰;
```

```
Sum += R;
```

```
a /= ۱۰;
```

```
}
```

```
MessageBox . show(sum.ToString());
```

:Do _ while ساختار

```
Int a=convert.ToInt32(txtnumber.text);
```

```
Int Sum =0;
```

```
Int R =0;
```

```
Do
```

```
{
```

```
R= a% ۱۰;
```

```
Sum += R;
```

```
a = a/۱۰;
```

```
}While(a>0);
```

```
MessageBox . show(sum.ToString());
```

ساختمان for :

برنامه ای بنویسید که ۲ عدد را از ورودی بگیرد و آن دو عدد و اعداد بین آنها را در خروجی نمایش دهد.

```
Int number¹=convert.ToInt³²(txtnum¹.Text);
```

```
Int number²=convert.ToInt³²(txtnum².Text);
```

```
String str="";
```

```
For ( int i = number¹ ; i <= number² ; i ++)
```

```
str = str + i.ToString() + ",";
```

```
MessageBox.show(str);
```

برنامه ای بنویسید که اعداد زوج ۰ تا ۱۰۰ را با استفاده از ساختمان for چاپ کند.

```
For ( int i = ۰ ; i <= ۱۰۰ ; i ++)
```

```
{ If (i%۲==۰)
```

```
MessaheBox.Show(i.ToString());
```

```
}
```

برنامه ای بنویسید که یک عدد را دریافت کند و فاکتوریل آن عدد را محاسبه و چاپ نماید.

```
Int num = convert.toInt³²(txtnum.text);
```

```
Int mult=۱;
```

```
For( int i=۱ ; i <= num ; i ++)
```

```
mult = mult * i ;
```

```
MessageBox.show(mult.toString());
```

ساختمان foreach :

یک حلقه است که برای جست و جو در آرایه به کار برده می شود. شکل کلی:

```
Foreach(int a in Array)
```

```
{ دستورات ;
```

```
}
```

مثال: برنامه ای بنویسید که مجموع عناصر عددی در یک آرایه را محاسبه و چاپ کند.

```
int[] a=new int[۵]{۳,۲۵,۱۰,۹۶,۷}; int sum=۰;
```

```
foreach(int b in a)
```

```
sum=sum+b;
```

```
MessageBox.Show(sum.ToString()); خروجی: ۱۴۱
```



```
int a;
a = Convert.ToInt32(txtnumber1.Text);
if (a % 2 == 0)
{
    MessageBox.Show("ZOD");
    MessageBox.Show(a.ToString());
}
else
{
    MessageBox.Show("FARD");
    MessageBox.Show(a.ToString());
}
-----

int a,b,max;
a = Convert.ToInt32(txtnumber1.Text);
b = Convert.ToInt32(txtnumber2.Text);
max = a;
if (max > b)

    MessageBox.Show(max.ToString());

else
{
    max = b;
    MessageBox.Show(max.ToString());
}
-----

int a,b,c,max;
a = Convert.ToInt32(txtnumber1.Text);
b = Convert.ToInt32(txtnumber2.Text);
c = Convert.ToInt32(txtnumber3.Text);
max = a;
if (max < b)
    max = b;
if (max < c)
    max = c;
    MessageBox.Show(max.ToString());
-----

int a;
a = Convert.ToInt32(txtnumber1.Text);
if(a == 1)
    MessageBox.Show("one");
else if (a == 2)
    MessageBox.Show("two");
else if(a == 3)
    MessageBox.Show("three");
else
    MessageBox.Show("Error!");
-----
```

```
int a;
a = Convert.ToInt32(txtnumber1.Text);
switch (a)
{
    case 1:
        MessageBox.Show("one");
        break;
    case 2:
        MessageBox.Show("two");
        break;
    case 3:
        MessageBox.Show("three");
        break;
    default:
        MessageBox.Show("Error!");
        break;
}

-----

string s = "";
s = System.DateTime.Now.DayOfWeek.ToString();
switch (s)
{
    case "Saturday":
        MessageBox.Show("شنبه");
        break;
    case "Sunday":
        MessageBox.Show("یکشنبه");
        break;
    case "Monday":
        MessageBox.Show("دوشنبه");
        break;
    default:
        MessageBox.Show("Error!");
        break;
}

-----

int a,sum,r;
sum=0;
a = Convert.ToInt32(txtnumber1.Text);
do
{
    r = a % 10;
    sum = sum + r;
    a = a / 10;
} while (a > 0);
MessageBox.Show(sum.ToString());
```

```
    }
}
}
```

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

namespace WindowsFormsApplication5
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        int sumnuber(int a)
        {
            int sum = 0;
            for (int i = 1; i <= a; i++)
                sum = sum + i;
            return sum;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            int sum = 0;
            for (int i = 0; i <= 10; i++)
                sum = sum + i;
            MessageBox.Show(sum.ToString());
            -----
            int[] A = new int[5] {20,5,12,18,2 };
            int sum = 0;
            for (int i = 0; i < A.Length; i++)
                sum = sum + A[i];
            MessageBox.Show(sum.ToString());
            -----
            int a = Convert.ToInt32(textBox1.Text);
            int b = Convert.ToInt32(textBox2.Text);
            string str = "";
            for (int i = a; i <= b; i++)
                if(i%2==0)
                    str = str + i.ToString() + ",";
            MessageBox.Show(str);
            -----
            int a = Convert.ToInt32(textBox1.Text);
            if (a % 3 == 0)
                MessageBox.Show("bakhshpazir bar 3 hast");
            else
                MessageBox.Show("bakhshpazir bar 3 nist");
            if (a % 5 == 0)
                MessageBox.Show("bakhshpazir bar 5 hast");
            else
                MessageBox.Show("bakhshpazir bar 5 nist");
            -----
            int n = Convert.ToInt32(textBox1.Text);
            int mult = 1;
```

```

for (int i = 1; i <= n; i++)
    mult = mult * i;
MessageBox.Show(mult.ToString());
-----
string str="";
for (int i = 100; i >= 1; i--)
    if (i % 2 != 0)
        str = str + i.ToString() + ",";
MessageBox.Show(str);
-----
int n = Convert.ToInt32(textBox1.Text);
int sum = 0;
for (int i = 1; i <= n; i++)
    sum = sum + i;
double k = sum + Math.Sin(n);
MessageBox.Show(k.ToString());
-----
int n = Convert.ToInt32(textBox1.Text);
double k = sumnuber(n) + Math.Sin(n);
MessageBox.Show(k.ToString());

    }
}
}

```

تعریف تابع (متد Method):

گاهی اوقات تعداد خطوط کدهای برنامه زیاد می شود و برای اینکه برنامه های بزرگ قابل مدیریت باشند و اشکال زدایی و رفع عیب آنها راحت تر صورت بگیرد ، برنامه نویسان این کدها را به بخش های کوچک تری با تعداد خطوط کمتر تقسیم می کنند که به این بخش ها تابع گفته می شود. و هر تابع دارای وظیفه مشخصی است.

توابع:

۱. توابع کتابخانه ای: به توابع پیش فرض برنامه که همراه با کامپایلر برنامه هستند، توابع کتابخانه ای می گویند. مانند تابع $\sin(a)$.
۲. توابع کاربر ساز: توابعی که می تواند دارای نوع بازگشتی باشند یا نباشند و توسط برنامه نویس (کاربر) نوشته می شود.

کلاس math یک کلاس شامل توابع ریاضیاتی است که با استفاده از این توابع، انجام محاسبات ریاضی برای برنامه نویسان آسان تر می شود.

برنامه ای بنویسید که یک عدد را از ورودی دریافت کند و حاصل جمع از ۱ تا آن عدد و مجموع آنها و سینوس آن عدد را محاسبه کند.

۳. Int num = convert.toint^{۳۲}(txtnum.text);
۴. Int sum = ۰;
۵. For(int i = ۱ ; i<= num ; i++)
۶. Sum= sum+ i;
۷. Double k=sum + math.sin(num);
۸. MessageBox.show(k.Tostring());

توابع کاربر ساز:

۱. توابعی که فقط یک مقدار بازگشتی (یا یک مقدار خروجی) دارند. مثال:

(پارامترها) نام تابع نوع خروجی تابع

```
{
Functions body
}
int sum(int num۱ , int num۲)
{ return num۱+num۲ ;
}
```

برای فراخوانی این نوع تابع از طریق زیر اقدام میکنیم:

```
int num۱ = int.parse(txt۱.Text); // مقدار عدد اول را از ورودی دریافت میکند
int num۲ = int.parse(txt۲.Text); // مقدار عدد دوم را از ورودی دریافت میکند
int sum۳ = sum(num۱ , num۲); // مقدار های دریافتی از ورودی را به تابع ارجاع میدهد و پس
از عملیات داخل تابع، نتیجه تابع، در نام تابع ذخیره می شود و سپس از نام تابع، داخل یک متغیر ریخته می شود.
MessageBox.show(sum۳.ToString());
```

۲. تابع بدون نوع بازگشتی (void):

در این نوع تابع مقداری بر نمی گردد پس در تابع به return نیازی نداریم

```
Void sum (int num۱ , int num۲)
{
int s = num۱+num۲ ;
messageBox.show(s.toString());
}
```

این نوع تابع را به شکل زیر فراخوانی می کنیم:

```
int num۱=int.parse(txt۱.Text);
int num۲=int.parse(txt۲.Text);
sum (num۱ , num۲);
```

نکته: نوع داده double میتواند یک عدد از نوع متغیری int را نگه دارد اما int نمیتواند داده double را نگه داری کند.

```
Void taghsim(int num۱ , int num۲)
{
```

```
double kharegghesmat=(double) num\ / num۲;
int baghimandeh= (double)num\ % num۲;
MessageBox.show("baghimandeh={۰} va kharegghesmat={۱}", baghimandeh , kharegghesmat);
}
```

این تابع را به شکل زیر فراخوانی میکنیم:

```
int a = int.parse(txt۱.Text); // مقدار عدد اول را از ورودی دریافت میکند
int b = int.parse(txt۲.Text); // مقدار عدد دوم را از ورودی دریافت میکند
taghsim (a, b);
```

۳. تابع خروجی با نوع متغیر ref (refrence) (نکته: در اینجا تابع بیش از یک مقدار خروجی دارد، یعنی هم در نام تابع و هم در متغیر ref).

نکته: در این نوع تابع، همچنین می توان به جای اینکه نتیجه را با استفاده از return برگردانیم، آن را در متغیری از نوع ref برگردانیم.

مثال:

```
Double taghsim(int num۱ , int num۲ , ref int baghimandeh)
{
double x = (double)num۱ / num۲;
Baghimandeh = num۱%num۲;
Return x;
}
```

این نوع تابع را به شکل زیر فراخوانی میکنیم:

```
int a = int.Parse(textBox۱.Text);
int b = int.Parse(textBox۲.Text);
int r = ۰; نکته: حتما متغیر ref قبل از ورود به تابع باید مقدار دهی اولیه بشود.
double d = taghsim(a, b, ref r);
MessageBox.Show(d.ToString());
MessageBox.Show(r.ToString());
```

چنانچه بخواهیم برنامه بالا را کاملا استاندارد بنویسیم باید به صورت زیر انجام دهیم.

```

bool Div(int a,int b, ref double res)
{
    if ( b != 0 )
    {
        res = Math.Round((double )a / b , 4);
        return true ;
    }
    else
        return false ;
}

1 reference
private void BtnDiv_Click(object sender, EventArgs e)
{
    int f = int.Parse(txtFirst.Text);
    int s = int.Parse(txtSecond.Text);
    double d=0;
    bool r = Div(f, s, ref d);
    if(r == false )
        MessageBox.Show("makhraj is zero");
    else
        MessageBox.Show(d.ToString());
}

```

۴. تابع با متغیر خروجی از نوع out :

همانند تابع نوع قبل است فقط به جای نام ref از نام out استفاده میکنیم.

نکته: این متغیر out در برنامه نیاز به مقدار دهی اولیه ندارد. مثال:

```

void taghsim\ (int aa, int bb, out double kh,out int rr)
{
    kh = (double)aa / bb;
    rr = aa % bb;
}

int a = int.Parse(textBox\ .Text);
int b = int.Parse(textBox\ .Text);
double k;
int r;
taghsim\ (a, b, out k,out r);
MessageBox.Show(k.ToString());
MessageBox.Show(r.ToString());

```

```

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

```

```
namespace WindowsFormsApplication6
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

نکته: در visualstudio میتوانیم با زدن /// قبل از نوشتن تابع ، یک خلاصه ای از عملکرد تابع و مقادیر داخل تابع بنویسیم.

دیدن بدنه یک تابع: انتخاب نام تابع -> راست کلیک روی آن -> انتخاب گزینه go to Definition

یا با زدن کلید F۱۲

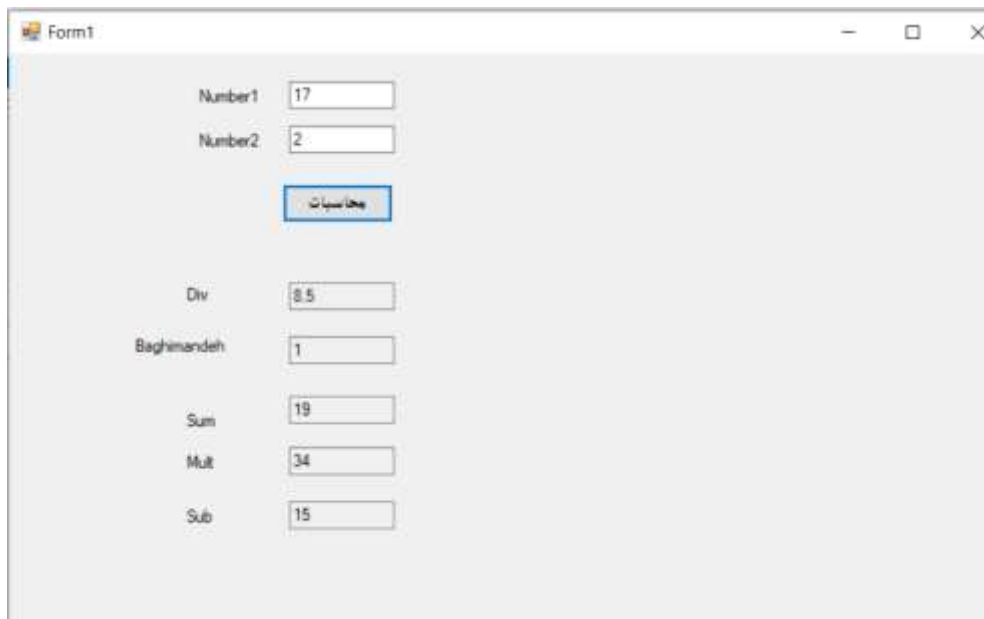
```
<summary>
    عدد دو مجموع محاسبه
</summary>
<param name="aa">عدد اول</param>
<param name="bb">عدد دوم</param>
<returns></returns>
int sum(int aa, int bb)
{
    return aa + bb;
}
int sum(int aa, int bb, int cc)
{
    return aa + bb + cc;
}
void sum(int aa, int bb)
{
    int s= aa + bb;
    MessageBox.Show(s.ToString());
}
double taghsim(int aa, int bb, ref int rr)
{
    double x=(double)aa / bb;
    rr = aa % bb;
    return x;
}
void taghsim1(int aa, int bb, out double kh,out int rr)
{
    kh = (double)aa / bb;
    rr = aa % bb;
}
private void button1_Click(object sender, EventArgs e)
{
    int[] a = new int[4] { 41, 20, 68, 100 };
    int sum=0;
    foreach(int b in a)
        sum=sum+b;
    MessageBox.Show(sum.ToString());
    -----
    int a = int.Parse(textBox1.Text);
    int b = int.Parse(textBox2.Text);
    int s = sum(a, b);
    MessageBox.Show(s.ToString());
    int a = int.Parse(textBox1.Text);
```



```
int b = int.Parse(textBox2.Text);
int c = int.Parse(textBox3.Text);
int s = sum(a, b, c);
MessageBox.Show(s.ToString());
MessageBox.Show("hell", "Error", MessageBoxButtons.YesNoCancel);
//-----
int a = int.Parse(textBox1.Text);
int b = int.Parse(textBox2.Text);
sum(a, b);
-----
int a = int.Parse(textBox1.Text);
int b = int.Parse(textBox2.Text);
int r = 0;
double d = taghsim(a, b, ref r);
MessageBox.Show(d.ToString());
MessageBox.Show(r.ToString());
-----
int a = int.Parse(textBox1.Text);
int b = int.Parse(textBox2.Text);
//double k;
int r;
taghsim1(a, b, out k, out r);
MessageBox.Show(k.ToString());
MessageBox.Show(r.ToString());
-----
//int a = int.Parse(textBox1.Text);

if(int.TryParse(textBox1.Text, int a)==false)
{
    MessageBox.Show("number aval invalid");
    return ;
}
int b = int.Parse(textBox2.Text);
int s = sum(a, b);
MessageBox.Show(s.ToString());
MessageBox.Show("number is invalid");

}
}
```



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

    bool taghsim(double aa1, double bb1, out double div11, out double baghi1)
    {
        if (bb1 != 0)
        {
            div11 = aa1 / bb1;
            baghi1 = aa1 % bb1;
            return true;
        }
        div11 = 0;
        baghi1 = 0;
        return false;
    }

    private void btnClculate_Click(object sender, EventArgs e)
    {
        if (txtNumber1.Text == "")
        {
            txtSum.Text = "";
            txtDiv.Text = "";
            MessageBox.Show("کنید وارد اول عدد لطفا");
            txtNumber1.Focus();
            return;
        }
        if (txtNumber2.Text == "")
        {
            txtSum.Text = "";
            txtDiv.Text = "";
            MessageBox.Show("کنید وارد دوم عدد لطفا");
            txtNumber2.Focus();
            return;
        }
    }
}
```

```

int aa;
if( int.TryParse(txtNumber1 .Text, out aa)==false)
{
    MessageBox.Show("است نامعتبر اول عدد");
    txtNumber1.Focus();
    return;
}
//int b = int.Parse(txtNumber2.Text);
int bb;
if (int.TryParse(txtNumber2.Text, out bb) == false)
{
    MessageBox.Show("است نامعتبر دوم عدد");
    txtNumber2.Focus();
    return;
}
double sum = aa + bb;
txtSum.Text = sum.ToString();
double sub = aa - bb;
txtSub.Text = sub.ToString();
double mult = aa * bb;
txtMult .Text = mult.ToString();
double div1;
double baghi;
bool kh = taghsim(aa, bb, out div1, out baghi);
if (kh == true)
{
    txtDiv.Text = div1.ToString();
    txtBaghimandeh.Text = baghi.ToString();
}
else
{
    MessageBox.Show("قسمت خارج=است صفر دوم عدد");
    return;
}
}
}

```

توابع overload :

توابعی هستند که دارای یک نام یکسان هستند، ولی میتوانند در تعداد پارامترها و یا نوع پارامتر ها متفاوت باشند، این توابع باعث خوانایی بیشتر برنامه می شوند.

مانند تابع Show در کلاس MessageBox که دارای ۲۱ حالت تابع است و از هر کدام در صورت نیاز میتوانیم استفاده کنیم.

مثل:

MessageBox.show("Message" , "title" , messageboxButton.yes/no/cancle);

نکته :

برای نمایش دادن overload های یک تابع میتوانیم از کلید های ctrl + shift + space استفاده کنیم.

ساختار Try & catch

از این ساختار برای کنترل خطا های زمان اجرا که قابل پیش بینی نیستند استفاده می کنیم.
شکل کلی:

Try

{

کد های برنامه

}

Catch

{

کد اجرایی بعد از برخورد به یک ارور

}

نکته: در ساختار try & catch می توانیم از catch ها با تعداد نامعلوم استفاده کنیم.
دستور finally بعد از catch نوشته می شود و در هر صورت دستورات داخل آن اجرا می شوند.

دستور try parse : کنترل خطای زمان اجرا با بکار بردن دستور try parse

(روش دیگر برای مدیریت ورودی ها است.)

مثال: برنامه ای بنویسید که دو عدد از ورودی دریافت کرده، جمع آنها را محاسبه و نمایش دهد. کنترل های لازم را انجام دهید.

```
private void BtnSum_Click(object sender, EventArgs e)
{
    //int i = int.Parse(txtFir.Text );
    //int j = int.Parse(txtSec.Text );

    if (int.TryParse(txtFir.Text, out int i) == false)
    {
        MessageBox.Show("number First is invalid");
        return;
    }
    if (int.TryParse(txtSec.Text, out int j) == false)
    {
        MessageBox.Show("number second is invalid");
        return;
    }
    MessageBox.Show((i + j).ToString());
}
```

بکار بردن دستور try catch

```
private void BtnSum_Click(object sender, EventArgs e)
{
    //int i = int.Parse(txtFir.Text );
    //int j = int.Parse(txtSec.Text );

    //if (int.TryParse(txtFir.Text, out int i) == false)
    //{
    //    MessageBox.Show("number First is invalid");
    //    return;
    //}
    if (cntrolTryparse(txtFir.Text, out int i) == false)
    {
        MessageBox.Show("number First is invalid");
        return;
    }
    if (int.TryParse(txtSec.Text, out int j) == false)
    {
        MessageBox.Show("number second is invalid");
        return;
    }
    MessageBox.Show((i + j).ToString());
}

bool cntrolTryparse(string s, out int i)
{
    try
    {
        i = int.Parse (s);
        return true;
    }
    catch
    {
        i = 0;
        return false;
    }
}
```

فرمتهای نمایش متن و محتوای متغیرها توسط تابع `MessageBox.show()`

```
public partial class Form1 : Form
{
    1 reference
    public Form1()
    {
        InitializeComponent();
    }

    1 reference
    private void Btnshow_Click(object sender, EventArgs e)
    {
        long p = long.Parse(txtPayment.Text);
        int r = Convert.ToInt32(txtReward.Text);
        //MessageBox.Show("hello" + " " + txtName.Text + " " + txtFamily.Text);
        //MessageBox.Show(string.Format("hello {0} {1} \n pay:{2 :N0}",txtName.Text, txtFamily.Text, p + r));
        MessageBox.Show($"hello {txtName.Text} {txtFamily.Text} \n pay: {(p+r).ToString("N0")}");
    }
}
```

دستورات کاربردی برای رشته ها: Length – Replace – Substring – IndexOf – LastIndexOf

مثال:

```
private void BtnLen_Click(object sender, EventArgs e)
{
    //int i= textBox1.Text.Length;
    //MessageBox.Show(i.ToString());
    MessageBox.Show(textBox1.Text.Length.ToString());
}

private void Relace_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox1.Text.Replace('a', '%');
}

private void Substring_Click(object sender, EventArgs e)
{
    string year=textBox1.Text.Substring(0, 4);
    string month= textBox1.Text.Substring(5, 2);
    string day = textBox1.Text.Substring(8);
    MessageBox.Show("year: " + year+"\n month:" + month + " \n day: " + day);
}

private void InexOf_Click(object sender, EventArgs e)
{
    int i = textBox1.Text.IndexOf('@');
    //MessageBox.Show(i.ToString());
    if (i > -1)

    MessageBox.Show(textBox1.Text.Substring(textBox1.Text.IndexOf('@') + 1));
    else
        MessageBox.Show("email is invalid");

    //MessageBox.Show(textBox1.Text.Substring(textBox1.Text.LastIndexOf()));
}

private void NameFile_Click(object sender, EventArgs e)
{
    int b = -1, last = -1;
    do
    {
        last = b;
        b = textBox1.Text.IndexOf('\\', b + 1);
    }
    while (b > -1);
    if (last>-1)
        MessageBox.Show(textBox1.Text.Substring(last + 1));
}
```

```
        else
            MessageBox.Show("File not Found");
    }

    private void LastindexOf_Click(object sender, EventArgs e)
    {
        MessageBox.Show(textBox1.Text.Substring(textBox1.Text.LastIndexOf('\\') +
        1));
    }
```

مقایسه رشته ها با دستور Compare

```
private void Compare_Click(object sender, EventArgs e)
{
    //int i=
    textBox1.Text.ToLower().CompareTo(textBox2.Text.ToLower());
    //MessageBox.Show(i.ToString());

    int i = string.Compare(textBox1.Text, textBox2.Text, true);
    if (i>0)
        MessageBox.Show("larger");
    else if (i<0)
        MessageBox.Show("smaller");
    else
        MessageBox.Show("equal");
}
```

تمرین:

با استفاده از توابع قبلی که یاد گرفتید یک تاریخ از کاربر دریافت کرده و آن را به شکل تاریخ استاندارد تبدیل کرده و نمایش دهید.

```
//تابع تاریخ
bool checkdate(string Date, out string Correctdate)
{
    Correctdate = "";
    int pos1 = Date.IndexOf('/'); int pos2 =
    Date.LastIndexOf('/');
    if (pos1 == -1 || pos1 == pos2)
        return false;
    string year = Date.Substring(0,pos1);
    string month = Date.Substring(pos1 + 1, pos2 - pos1 - 1);
    string day = Date.Substring(pos2+1);
    if (int.TryParse(year, out int y) == false)
        return false;
    if (!int.TryParse(month, out int m))
        return false;
    if (!int.TryParse(day,out int d))
        return false;
```



```

        if (y >= 0 && y < 100)
            y += 1300;
        if (y > 1400)
            return false;
        if (m < 1 || m > 12)
            return false;
        if (m < 7 && d > 31)
            return false;
        if (m > 7 && d > 30)
            return false;
        if (y % 4 == 3 && m == 12 && d > 30)
            return false;
        if (y % 4 != 3 && m == 12 && d > 29)
            return false;

        //if ((m>7 && d>30) || (y%4==3 && m==12 && d>30) || (y%4!=3&&
m==12 && d>29))
            //return false;

        Correctdate = string.Format("{0} /{1:00} /{2:00}", y, m, d);
        return true;
    }

    فراخوانی تابع تاریخ
    private void Date_Click(object sender, EventArgs e)
    {
        string cd;
        if( checkdate(textBox1.Text, out cd))
            textBox1.Text = cd;
        else
            MessageBox.Show(" تاریخ نامعتبر است ");
    }

```

توابع تبدیل تاریخ(میلادی به شمسی و برعکس)

تاریخ و ساعت میلادی جاری سیستم(فعلي):

```
private void Button1_Click(object sender, EventArgs e)
{
    DateTime dt = DateTime.Now;
    MessageBox.Show(dt.ToString ("yyyy/MM/dd - HH:mm"));
}
```

تاریخ و ساعت میلادی مورد نظر خودمان:

```
DateTime dt2 = new DateTime(2020, 1, 1);
MessageBox.Show(dt2.ToString("yyyy/MM/dd - HH:mm"));
```

دیدن فاصله زمانی (برحسب روز. ساعت. دقیقه) از الان تا تاریخ تولدمان:

```
private void Button1_Click(object sender, EventArgs e)
{
    DateTime dt = DateTime.Now;
    DateTime dt2 = new DateTime(2020, 1, 1);
    TimeSpan ts = dt - dt2;
    MessageBox.Show(ts.Days.ToString());
}
```

میخواهیم تاریخ هفته بعد را بگیریم (۷) و هفته قبل (۷-)

```
DateTime dt = DateTime.Now;
dt = dt.AddDays(7);
MessageBox.Show(dt.ToString ());
```

تبدیل تاریخ فعلی میلادی به تاریخ شمسی با استفاده از کلاس **:PersianCalendar**

```
private void datefelimiladitoShamsi_Click(object sender, EventArgs e)
{
    DateTime dt = DateTime.Now;
    PersianCalendar pc = new PersianCalendar();
    string Date =
    $"{pc.GetYear(dt)}/{pc.GetMonth(dt).ToString("00")}/{pc.GetDayOfMont
h(dt).ToString ("00")} ";
```

```
        MessageBox.Show(Date);  
    }
```

تبدیل تاریخ مورد نظر میلادی به تاریخ شمسی با استفاده از کلاس **PersianCalendar**:

```
using System.Globalization;  
private void miladitoShamsi_Click(object sender, EventArgs e)  
{  
    //DateTime dt = DateTime.Now;  
    DateTime dt2 = new DateTime(2020,1,1);  
    PersianCalendar pc = new PersianCalendar();  
    string Date =  
    $"{pc.GetYear(dt2)}/{pc.GetMonth(dt2).ToString("00")}/{pc.GetDayOfMo  
nth(dt2).ToString("00")} ";  
    MessageBox.Show(Date);  
    -----  
    DateTime dt = DateTime.Now;  
    DateTime dt2 = new DateTime(2020, 06, 25);  
    System.Globalization.PersianCalendar pc = new  
    System.Globalization.PersianCalendar();  
    string Date =  
    string.Format("{0}/{1:00}/{2:00}", pc.GetYear(dt).ToString(), pc.GetMo  
nth(dt), pc.GetDayOfMonth(dt));  
    MessageBox.Show(Date);  
}
```

تبدیل تاریخ مورد نظر شمسی به تاریخ میلادی با استفاده از کلاس **PersianCalendar**:

```
private void Shamsitomiladi_Click(object sender, EventArgs e)  
{  
    System.Globalization.PersianCalendar pc = new  
    System.Globalization.PersianCalendar();  
    DateTime dt = new DateTime(1401,04,05,pc);  
    MessageBox.Show(dt.ToString("yyyy/MM/dd"));  
}
```

تمرین: برنامه ای بنویسید که از ورودی تاریخ میلادی گرفته به شمسی تبدیل کند یا برعکس.
تمرین: برنامه ای بنویسید که تاریخ تولد کاربر را از ورودی گرفته، سن او را به او بگوید.

مفاهيم مربوط به کلاس

کلاس (Class) شیء (Object)

یکی از ساختارهای مطرح در هر زبان برنامه نویسی، کلاس می باشد. هر کلاس در واقع محدوده ای است که داده ها به همراه عملیات های تعریف شده روی آن ها پیاده سازی می شود. هر کلاس از دو منظر مورد اهمیت قرار دارد :

□ بحث: Classification وقتی ما داده ها را در قالب یک کلاس مطرح می نماییم به طور سیستمی رفتار یک موجودیت (entity) را پیاده سازی می نماییم .

□ بحث: Encapsulation با تعریف یک کلاس، دیگر مهم نیست که داخل کلاس، رفتارها چگونه پیاده سازی شده است و چگونه عمل می کند. ما تنها با فرخوانی متدهای کلاس از آن استفاده می نماییم و در واقع عملیات های درونی کلاس از برنامه هایی که از آن کلاس استفاده می کنند، پنهان می ماند .

فرم کلی تعریف کلاس به شکل زیر می باشد :

```

نام کلاس  class
{
    (تعریف ویژگی های کلاس) Fields;

    -----
    -----
    -----

    (تعریف متدهای کلاس) Methods;

    -----
    -----
    -----

}

```

:Tip

□ وقتی از یک کلاس یک نسخه ایجاد می نماییم، در واقع یک شیء (object) از کلاس ساخته ایم. برای ساخت object از کلاس از کلمه new استفاده می شود .

new نام کلاس = نام شیء نام کلاس ();

کلاس در واقع یک نوع (Type) ایجاد می نماید و object یک نمونه از کلاس است .

□ بر خلاف تعریف متغیرها، در صورتی که فیلدها داخل کلاس مقدار اولیه نگیرند بر حسب نوعشان یکی از مقادیر صفر، false ، یا null را می پذیرند .

□ هر عنصر در کلاس توسط یک از دو کلمه private یا public تعریف می شود. این مقادیر سطح دسترسی به عنصر را مشخص می نماید .

□ عناصر private : فیلدها و متدهایی که چنین خاصیتی دارند از بیرون از کلاس قابل دسترسی نیستند .

□ عناصر public : فیلدها و متدهایی که چنین خاصیتی دارند از بیرون از کلاس هم قابل دسترسی می باشند .

□ اگر نوع فیلد یا متدی از کلاس تعریف نشود، آن فیلد یا متد از نوع private در نظر گرفته می شود .

□ اصولاً برای نامگذاری موارد مختلف در کلاس استانداردهای زیر پیشنهاد می شود :

□ در تعریف متدها و فیلدهایی از کلاس که public تعریف می شوند pascal case notation

□ در تعریف متدها و فیلدهایی از کلاس که private تعریف می شوند camel case notation

□ نام کلاس خود pascal case notation

□ نام شی (object) های یک کلاس camel case notation

مثال: با ارائه مثالی تمام مفاهیم ارائه شده را با هم بررسی می نماییم .

کلاسی به نام Box تعریف نمایید. این کلاس در واقع یک مکعب مستطیل را پیاده سازی می کند. در این کلاس

طول tol ، عرض arz و ارتفاع ertefa مکعب مستطیل تعریف می شود. همچنین این کلاس

قابلیت محاسبه مساحت قاعده مکعب مستطیل و حجم آن را خواهد داشت .

مساحت قاعده = طول * عرض

حجم مکعب مستطیل = طول * عرض * ارتفاع

برای ایجاد و اضافه نمودن یک کلاس به پروژه کافی است که از یکی از مسیرهای زیر استفاده نمایید :

Solution Explorer □ Add □ class □ نام کلاس □ Box.cs راست کلیک روی نام پروژه □

□ Project □ Add class □ نام کلاس □ Box.cs یا منو نوار در

کلاس را به فرم زیر تعریف می نماییم :

```
namespace WindowsFormsApp9
{
    class Box
    {
        private float tol;
```

```

private float arz;
private float ertefa;

public void settolarzertefa(float t, float a, float e)
{
    tol = t;
    arz = a;
    ertefa = e;
}

public float masahatghaedehe(float tt, float aa)
{
    tol = tt;
    arz = aa;
    return tol * arz;
}

public float hajm()
{
    return tol * arz * ertefa;
}
}

```

□ همانطور که مشاهده می نمایید متدها و فیلدهای public به روش pascal case notation

نامگذاری شده است و بقیه عناصر به روش Camel case notation نامگذاری شده است.

□ متد () masahatghaedehe ، مساحت قاعده مکعب مستطیل و متد hajm() ، حجم آن را محاسبه می نماید.

□ سه فیلد tol ، arz ، و ertefa به صورت private تعریف شده اند. بنابراین از خارج از کلاس به صورت مستقیم به آن ها دسترسی نداریم .

حال در پروژه خود فرمی به صورت زیر طراحی نمایید :

قرار است با ورود سه مقدار اول و کلیک روی دکمه محاسبات ، یک مکعب مستطیل ایجاد شود و مساحت قاعده

و حجم آن محاسبه و در فیلدهای مساحت و حجم قرار گیرد .

```
using System;
```

```
using System.Windows.Forms;
```

```
namespace WindowsFormsApp9
```

```
{
```

```
    public partial class Form3 : Form
```

```
    {
```

```
        public Form3()
```

```
        {
```

```
            InitializeComponent();
```

```
        }
```

```
        private void BtnClculate_Click(object sender, EventArgs e)
```

```
        {
```

```
            float t1, a1, e1, masahat1, hajm1;
```

```
            if (txtTol.Text == "")
```

```
            {
```

```
                txtMasahat.Text = "";
```

```
                txtHajm.Text = "";
```

```
                MessageBox.Show("کنید وارد طول برای عدد یک لطفا");
```

```
                txtTol.Focus();
```

```
                return;
```

```
            }
```

```
            if (txtArz.Text == "")
```

```
            {
```

```
                txtMasahat.Text = "";
```

```
                txtHajm.Text = "";
```

```
                MessageBox.Show("کنید وارد عرض برای عدد یک لطفا");
```

```
                txtArz.Focus();
```

```
                return;
```

```
            }
```

```
            if (txtErtefa.Text == "")
```

```
            {
```

```
                txtMasahat.Text = "";
```

```
                txtHajm.Text = "";
```

```
                MessageBox.Show("کنید وارد ارتفاع برای عدد یک لطفا");
```

```
                txtErtefa.Focus();
```

```
                return;
```

```
            }
```

```
            t1 = int.Parse(txtTol.Text);
```

```
            a1 = int.Parse(txtArz.Text);
```

```
            e1 = int.Parse(txtErtefa.Text);
```

```
            Box box = new Box();
```

```
            // گردد دهی مقدار جا همین از خصوصیت بودن پابلیک صورت
```

```
            // box.ertefta = e1;
```

```
            box.settolarzertefa(t1, a1, e1);
```

```
            masahat1 = box.masahatghaedeht1, a1);
```

```
            hajm1 = box.hajm();
```

```
            txtMasahat.Text = masahat1.ToString();
```

```

txtHajm.Text = hajm1.ToString();
//-----
// Class1 clas = new Class1(8,7);
////Class1 clas = new Class1();
// MessageBox.Show((clas.a).ToString()); //8 داد نشان را
// MessageBox.Show((clas.mult()).ToString()); //56 داد نشان را
}
}
}

```

:Tip

□ همانطور که قبلاً مطرح شد سه فیلد tol، arz، و ertefa در کلاس از نوع private هستند.

بنابراین در کد بالا امکان نوشته box.tol وجود ندارد. این فیلد از خارج از کلاس قابل دسترس نخواهد بود.

مفهوم Constructor

برای استفاده از یک کلاس می بایست یک شیء (object) از نوع کلاس ایجاد نمود. زمانی که شما از کلمه new استفاده می نمایید، در زمان اجرا یک شیء از ساختار تعریف شده برای کلاس ایجاد می شود؛ به این صورت که یک فضای حافظه از سیستم عامل گرفته می شود، این فضا با فضای موردنیاز برای فیلدهای تعریف شده در کلاس پر می شود و سپس یک متد با نام متد سازنده (Method Constructor) برای مقداردهی اولیه به فیلدهای شیء نیاز می باشد. متد سازنده (Constructor) متدی همنام کلاس می باشد که به طور اتوماتیک وقتی شما یک نمونه از کلاس را ایجاد می نمایید (new) اجرا می شود. هر کلاس میبایست یک متد سازنده داشته باشد و اگر این متد برای کلاس تعریف نشود، کامپایلر خود یک متد سازنده پیش فرض (Constructor Default) برای کلاس ایجاد می نماید.

:Tip

□ در صورتی که فیلدها داخل کلاس مقدار اولیه نگیرند، بر حسب نوعشان یکی از مقادیر صفر، false، یا null را می پذیرند.

این دقیقاً کاری است که Method Constructor Default انجام می دهد. در صورتی که شما متد سازنده در کلاس تعریف نکرده باشید، کامپایلر یک Constructor default ایجاد می نماید که مقداردهی اولیه فیلدها را انجام می دهد.

برای ایجاد یک Constructor تنها کافی است که یک متد از نوع public هم نام با نام کلاس تعریف نمایید.

دقت نمایید که متدهای سازنده هیچ مقدار خروجی حتی void را بر نمی گردانند.

کلاس Box را به خاطر دارید. کافی است متد Box() را به فرم زیر به کلاس اضافه نمایید تا یک default Constructor داشته باشید.

```

namespace Session9_ClassWindowsFormsApp
{
    class Box
    {
        private float tol;

```



```

private float arz;

private float ertefa;

//Default Constructor Method

public Box()

{

    tol = ۸;

    arz = ۴;

    ertefa = ۲;

}

public float hajm()
{
    return tol * arz * ertefa;
}

}

```

□ default Constructor : یک متد سازنده است که هیچ پارامتر ورودی نمی گیرد و اهمیتی ندارد که شما آن را به کلاس اضافه نموده اید و (داخلش به فیلدها مقدار دهی اولیه داده اید) یا توسط کامپایلر به کلاس بصورت ضمنی اضافه شده است و (داخلش به فیلدها مقدار صفر داده شده است).

□ non - default Constructor : متد سازنده ای که دارای پارامترهای ورودی می باشد .

در تکه کد بالا برای کلاس Box یک Constructor default ایجاد شد بنابراین اگر دستوری به فرم زیر نوشته شود :

```
Box box = new Box();
```

با اجرای این دستور یک شی به نام box از جنس کلاس Box ایجاد می شود و با توجه به Constructor default

تعریف شده برای این کلاس مقادیر box.tol ، box.arz و box.ertefa همگی برابر مقادیر داده شده در متد خواهد بود .

مثال: کلاس Box را بار دیگر ایجاد نمایید. برای این کلاس یک non - default Constructor تعریف نمایید .

```

namespace Session۹_ClassWindowsFormsApp
{

    class Box

    {

        private float tol;

```

```

private float arz;

private float ertefa;

//non-Default Constructor Method

public Box(float t, float a, float e)
{
    tol = t;
    arz = a;
    ertefa = e;
}

public float hajm()
{
    return tol * arz * ertefa;
}

}

```

همانطور که مشاهده می نمایید در کلاس Box یک non - default Constructor تعریف گردید. بنابراین اگر دستوری به فرم زیر نوشته شود .

```
Box box = new Box(۵, ۷, ۲);
```

با اجرای این دستور یک شی با نام box از جنس کلاس Box ایجاد می شود و با توجه به non - default Constructor تعریف شده برای این کلاس مقادیر box.tol، box.arz، box.ertefa به ترتیب برابر ۷، ۵ و ۲ می شود .

کلاس

ایجاد کلاس Car

```

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApp18
{
    class Car
    {
        private int _charkhCount, _DoorCount, _speed;
        public Color color;
        public int charkhCount

```

```
{
    get
    {
        return _charkhCount;
    }
    set
    {
        if (value >= 2 && value <= 6)
            _charkhCount = value;
    }
}

public int DoorCount
{
    get
    {
        return _DoorCount;
    }
    set
    {
        if (value >= 2 && value <= 4)
            _DoorCount = value;
    }
}

public int speed
{
    get
    {
        return _speed;
    }
}

public Car()
{
    charkhCount = 4;
    DoorCount = 4;
    color = Color.White;
}

public Car(int charkhCount1, int doorCount)
{
    charkhCount = charkhCount1;
    DoorCount = doorCount;
}
```

```
        color = Color.White;
    }

    public Car(int charkhCount1, int doorCount, Color col)
    {
        charkhCount = charkhCount1;
        DoorCount = doorCount;
        color = col;
    }

    public int dormotor()
    {
        return charkhCount * DoorCount;
    }

    public int speedd()
    {
        return _speed += 10;
    }
}
}
```

در فرم ۱

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

namespace WindowsFormsApp18
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Button1_Click(object sender, EventArgs e)
        {

```

```

        Car c = new Car(3,3,Color.Pink);
        //c.charkhCount = 3;
        //c.DoorCount = 2;
        //c.color = Color.Red;
        // c.speed = 200;
        c.speedd();
        c.speedd();
        showinfocar(c);
    }

    private void showinfocar(Car car)
    {
        MessageBox.Show($"charkhCount :{car.charkhCount} \n
DoorCount" +
                        $" :{car.DoorCount}\ncolor: {car.color}\n dormotor:
{car.dormotor()}\n speed: {car.speed}");
    }
}

```

Vanet ایجاد کلاس

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace WindowsFormsApp18
{
    class vanet:Car
    {
        private int _wheight;
        public int wheight
        {
            get {
                return 20;
            }
        }
        public override int dormotor()
        {
            //return base.dormotor()*2;
            return DoorCount * charkhCount * 2;
        }
        public override int speedd()
        {
            return _speed += 5;
        }
    }
}

```

Car تغییرات کلاس

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApp18
{
    class Car
    {
        private int _charkhCount, _DoorCount;
        protected int _speed;
        public Color color;
        public int charkhCount
        {
            get
            {
                return _charkhCount;
            }
            set
            {
                if (value >= 2 && value <= 6)
                    _charkhCount = value;
            }
        }
        public int DoorCount
        {
            get
            {
                return _DoorCount;
            }
            set
            {
                if (value >= 2 && value <= 4)
                    _DoorCount = value;
            }
        }
        public int speed
        {
            get
            {
                return _speed;
            }
        }
        public Car()
        {
            charkhCount = 3;
            DoorCount = 4;
            color = Color.White;
        }
        public Car(int charkhCount1, int doorCount)
        {
            charkhCount = charkhCount1;
            DoorCount = doorCount;
            color = Color.White;
        }
    }
}
```

```

    }
    public Car(int charkhCount1, int doorCount, Color col)
    {
        charkhCount = charkhCount1;
        DoorCount = doorCount;
        color = col;
    }

    public virtual int dormotor()
    {
        return charkhCount * DoorCount;
    }
    public virtual int speedd()
    {
        return _speed += 10;
    }
}
}

```

تغییرات در فرم ۱

```

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

namespace WindowsFormsApp18
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Button1_Click(object sender, EventArgs e)
        {
            Car c = new Car(3,3,Color.Pink);
            //c.charkhCount = 3;
            //c.DoorCount = 2;
            //c.color = Color.Red;
            // c.speed = 200;
            c.speedd();
            c.speedd();
            showinfocar(c);
        }

        private void showinfocar(Car car)
        {
            MessageBox.Show($"charkhCount :{car.charkhCount} \n DoorCount" +
                $" :{car.DoorCount}\ncolor: {car.color}\n dormotor:"
                {car.dormotor()}\n" +
                $" speed: {car.speed}");
        }
    }
}

```

```

    }

    private void Button2_Click(object sender, EventArgs e)
    {
        vanet v = new vanet();
        // v.wheight;
        //v.speedd();
        v.speedd();
        //v.speedd();
        showinfocar(v);
    }
}

```

گامها:

- ۱- ایجاد کلاس Vanet ۲- ارث بری کلاس Vanet از کلاس Car ۳- گذاشتن دکمه Vanet روی فرم ۱
- ۱- نمونه ساختن از کلاس Vanet، داخل دکمه Vanet
- ۲- جا زدن کلاس V بجای کلاس Car در تابع showinfocar(v); به این عمل پلی مورفیسم یا چند ریختی می گویند.
- ۳- هر کلاسی به تابع showinfocar(V) داده شود توابع همان کلاس فراخوانی می شود. حال کلاس V یا کلاس Car باشد. (نکته بسیار مهم: در زمان تعریف خود تابع، پارامتر ورودیش را از نوع کار معرفی کردیم. پس دقیقاً دنبال خصوصیات و توابع در کلاس اصلی کار می کردد. و فقط آنها را نمایش می دهد اگر مال کلاس کار باشد که اوکیه. مال کلاس وانت هم override شده و اوکیه. ولی توابع جدید در کلاس وانت را نشون نمیده، چون پارامتر ورودی فقط در حد کلاس کار)
- نکته: اول اضافه شدن کلمه virtual در کلاس Car (کلاس پدر) ، سپس override کردن توابع در کلاس Vanet(کلاس فرزند- جهت تغییر دستورات تابع مورد نظر)
- نکته: اول protected int _speed; در کلاس Car بعد برطرف شدن Error در تابع speedd() کلاس Vanet
- نکته: وقتی یک کلاسی از کلاس دیگر ارث بری می کند می تواند به اشیاء public و protected کلاس Base(کلاس پدر) دسترسی داشته باشد. کلاس Car کلاس Base(کلاس پدر) است و کلاس Vanet از آن ارث بری کرده است.