



آشنایی با انواع داده‌ها و متغیرها

برنامه‌هایی که تاکنون نوشته‌ایم، به نشان دادن یک پیام یا حاصل یک عبارت بر روی صفحه نمایش محدود می‌شد، اما در برنامه‌های کاربردی با داده‌ها و مقادیر مختلف سروکار داریم و باید بر روی آن عملیاتی را انجام دهیم. بعضی از این مقادیر مانند تاریخ تولد یک شخص یا نمره یک دانش‌آموز از قبل مشخص نیستند. مقدار این نوع داده‌ها باید در هنگام اجرای برنامه، ابتدا از کاربر دریافت شوند و در مکانی از حافظه کامپیوتر نگهداری شوند و در ادامه برنامه و در جریان پردازش، مورد استفاده قرار گیرند. چه حافظه‌ای برای نگهداری داده‌ها در هنگام پردازش مناسب است؟

در این فصل با متغیرها آشنا می‌شویم که برای نگهداری موقتی داده‌ها در برنامه مورد استفاده قرار می‌گیرند. همچنین برای نگهداری اطلاعات و نمایش آنها بر روی صفحه نمایش از متغیرها استفاده می‌کنیم.

پس از پایان این فصل انتظار می‌رود که فراگیر بتواند:

- ◀ متغیر را تعریف کند و انواع متغیر را در برنامه‌های خود به کاربندد.
- ◀ انواع داده‌ها را نام ببرد و تفاوت کاربرد هر یک را توضیح دهد.
- ◀ میزان حافظه و محدوده انواع داده‌ها را بیان کند.
- ◀ متغیرها را به‌طور صحیح در برنامه اعلان کند و آن‌ها را مقداردهی نماید.
- ◀ شکل نمایش نقطه‌شمار را توضیح دهد و اعداد اعشاری را در این قالب بنویسد.
- ◀ از متد `ReadLine()` برای دریافت داده‌های یک برنامه از ورودی استفاده کند.
- ◀ بر روی رشته دریافتی از ورودی، تغییراتی داده و سپس نمایش دهد.
- ◀ از متد `Parse()` برای تبدیل یک رشته به یک عدد استفاده کند.

۱-۴- متغیر چیست؟

در هر کامپیوتر حافظه‌های مختلفی وجود دارد که هر یک برای انجام کار خاصی پیش‌بینی شده است. یک نوع از حافظه کامپیوتر که قادر است داده‌ها را نگهداری کند و به سرعت قابل دسترسی است، حافظه موقتی RAM^۱ است. از اطلاعات درون حافظه RAM، در هر لحظه می‌توان با اطلاع شد و یا در صورت لزوم محتویات آن را تغییر داد یا مقدار جدیدی را در آن ذخیره کرد.

با توجه به مطالب گفته شده، لازم است در یک برنامه، یک یا چند مکان (بایت) از حافظه RAM کامپیوتر برای نگهداری موقتی داده‌ها یا نتایج حاصل از پردازش مورد استفاده قرار گیرد. در زبان‌های برنامه‌نویسی به این مکان‌ها، متغیر^۲ گفته می‌شود زیرا می‌توان محتوای آنها را در طول اجرای برنامه تغییر داد.

نکته

متغیر: مکانی از حافظه RAM کامپیوتر است که برای نگهداری موقتی داده‌ها یا اطلاعات استفاده می‌شود.

متغیر را مانند یک ظرف در نظر بگیرید. در آشپزخانه ظروف متعددی با شکل ظاهری، اندازه و جنس مختلفی وجود دارد که هر یک برای نگهداری یک نوع غذا یا مایعات استفاده می‌شود که گنجایش آن را داشته باشد. در یک برنامه نیز برای نگهداری هر یک از داده‌ها با توجه به نوع و بزرگی داده، باید از متغیر مناسبی استفاده کنیم که بتواند داده را نگهداری کند.

۲-۴- روش اعلان (تعریف) و ایجاد متغیرها

قبل از اینکه بتوانید مقداری را در یک متغیر ذخیره کنید باید متغیری را ایجاد کنید که قادر باشد آن مقدار را به درستی ذخیره نماید. در هنگام ایجاد متغیر باید نوع متغیر را مشخص نمایید. در زبان C# برای ایجاد و مشخص کردن نوع متغیر، از الگوی زیر استفاده می‌شود.

نام متغیر نوع داده

دستور زیر را در نظر بگیرید:

```
int a;
```

۱- Random Access Memory

۲- Variable

در این دستور متغیر a از نوع عدد صحیح اعلان می‌شود. کلمه int نوع متغیر را مشخص می‌کند که قادر است اعداد صحیح را در خود نگهداری کند و a نام متغیر است. نام متغیر به وسیله برنامه نویس انتخاب می‌شود که بهتر است نام و نوع آن مطابق با داده‌ای باشد که مقداردهی می‌شود.

۳-۴- نوع داده^۱ (نوع متغیر)

نوع متغیر به طور کلی ۳ ویژگی را مشخص می‌کند :

- ۱- گنجایش یا ظرفیت متغیر : مثلاً نوع int چهار بایت است.
- ۲- نوع اطلاعاتی که در متغیر می‌توان ذخیره کرد : مثلاً در متغیر نوع int فقط اعداد صحیح و بدون ممیز قابل نگهداری است.
- ۳- چه عملیاتی را می‌توان بر روی آن انجام داد : مثلاً عملیات ریاضی معمول را می‌توان روی اعداد نوع int انجام داد.

در زبان C# علاوه بر نوع داده int، انواع دیگری از داده‌ها نیز دسته بندی و گروه بندی شده‌اند و نحوه نمایش یا نگهداری^۲ آنها در حافظه و عملیاتی که می‌توان بر روی آنها انجام داد از قبل مشخص و تعریف شده است و برای هر دسته یا گروه از داده‌ها، یک نام انتخاب شده است که به آن نوع داده اولیه^۳ یا درون ساخته می‌گویند. جدول ۱-۴ انواع داده و مشخصات هر یک را نشان می‌دهد.

برای مثال در جدول ۱-۴ نوع داده sbyte را در نظر بگیرید. این نوع داده، اعداد صحیح و بدون ممیز در محدوده ۱۲۸- تا ۱۲۷+ را شامل می‌شود که یک بایت حافظه را اشغال می‌کند و بر روی آنها می‌توان عملیات ریاضی را انجام داد. اگر در یک برنامه، متغیری از نوع sbyte را استفاده کنیم، قادر خواهیم بود به عنوان مثال عدد ۷۸ را در آن ذخیره کنیم. اما نمی‌توان عدد ۲۰۰ و یا عدد ۱/۵ را در آن نگهداری کرد. همچنین نوع داده byte اعداد صحیحی فقط در محدوده ۰ تا ۲۵۵ را شامل می‌شود که در یک بایت قرار می‌گیرد. در این نوع داده فقط اعداد مثبت یا بدون علامت^۴ قابل نمایش می‌باشند.

۱ – Data Type

۲ – Representation

۳ – Primitive Data Type or Built- In Data Type

۴ – Unsigned numbers

جدول ۱-۴- انواع داده

نوع داده	کاربرد نوع داده	مقدار حافظه (بایت)	کمترین مقدار	بیشترین مقدار
sbyte	اعداد صحیح	1	-128	127
byte	اعداد صحیح مثبت	1	0	255
short	اعداد صحیح	2	-32768	32767
ushort	اعداد صحیح مثبت	2	0	65535
int	اعداد صحیح	4	-2147483648	2147483647
uint	اعداد صحیح مثبت	4	0	4294967295
long	اعداد صحیح	8	-9223372036854778508	9223372036854778507
ulong	اعداد صحیح مثبت	8	0	18446744073709551615
float	اعداد اعشاری	4	-3.402823×10^{38}	3.402823×10^{38}
double	اعداد اعشاری با دقت زیاد	8	$-1.79769313486232 \times 10^{308}$	$1.79769313486232 \times 10^{308}$
decimal	اعداد صحیح بزرگ اعداد اعشاری با دقت بسیار زیاد	16	$-79228162514264337593543950335$ -7.9×10^{28}	$79228162514264337593543950335$ $+7.9 \times 10^{28}$
bool	مقدار منطقی	1	false	true
char	یک حرف یا علامت (کراکتر)	2	0 کد کراکتر مطابق با سیستم Unicode	65535 کد کراکتر مطابق با سیستم Unicode
string	رشته		-	-
object	آدرس یک داده		-	-

عددی

غیر عددی

دستور `byte age` ; متغیری به نام `age` ایجاد می‌کند که این متغیر بسیار کوچک و به ظرفیت یک بایت است و می‌تواند یکی از اعداد صفر تا ۲۵۵ را در خود ذخیره کند.

اگر بخواهید چند متغیر از یک نوع را تعریف کنید کافی است بعد از ذکر نوع داده، نام متغیرها را با علامت ویرگول از یکدیگر جدا کنید. مثلاً برای تعریف دو متغیر برای نگهداری حداقل و حداکثر درجه حرارت از دستور زیر استفاده می‌کنیم :

`sbyte minTemp , maxTemp ;`

نکته

اگر متغیر مقدار دهی نشود خطا می‌گیرد. هر نوع داده، مجموعه‌ای از مقادیر به همراه مجموعه‌ای از عملیات را مشخص می‌کند.

برای اعداد صحیح و بدون ممیز نوع داده‌های زیر استفاده می‌شود:

`sbyte`, `byte`, `short`, `ushort`, `int`, `uint`, `long`, `ulong`

و برای اعداد اعشاری می‌توانید از نوع داده‌های `float` و `double` استفاده کنید. نوع داده `float` برای اعداد اعشاری با دقت حداکثر ۷ رقم استفاده می‌شود. در صورتی که ارقام عدد بیش از آن باشد عدد گرد می‌شود. مثلاً عدد ۱۲۳/۴۵۶۷۸۹ به صورت عدد ۱۲۳/۴۵۶۸ قابل نگهداری است. نوع داده `double` برای اعداد اعشاری بسیار بزرگ و یا بسیار کوچک مانند جرم و بار الکتریکی یک الکترون و با دقت زیاد ۱۵ رقم استفاده می‌شود.

نکته



در زبان برنامه‌نویسی C#، قبل از اینکه بتوانید داده‌ای را در یک متغیر ذخیره

کنید باید متغیر را ایجاد (یا اعلان) کنید و در هنگام ایجاد کردن یک متغیر، باید نوع متغیر (نوع داده) را مشخص نمایید. مثال: `float mark`;

۴-۲- مقداردهی متغیرها

پس از تعریف یا ایجاد متغیر، می‌توانید در آن، مقداری را با توجه به نوع متغیر ذخیره کنید. توجه داشته باشید که در یک متغیر همواره فقط یک مقدار نگهداری می‌شود و با ذخیره کردن داده جدید در یک متغیر، مقدار قبلی آن از بین می‌رود. مقداردهی متغیرها به چند روش صورت می‌گیرد. با دستور زیر مستقیماً مقداری در متغیر قرار می‌گیرد به این دستور، دستور انتساب^۱ می‌گویند.

مقدار = نام متغیر;

`byte age`;

`age = 16`;

متغیر `age` از نوع عدد صحیح اعلان شده و با عدد ۱۶ مقداردهی شده است.

در هنگام تعریف یا ایجاد متغیر نیز می‌توانید آن را مستقیماً مقداردهی کنید که به آن مقداردهی اولیه می‌گویند. الگوی آن چنین است :

مقدار = نام متغیر نوع داده

بنابراین دو دستور قبل را با الگوی بالا جایگزین می‌کنیم :

```
byte age = 16 ;
```

نکته

- ۱- برای مشخص کردن اعداد مثبت نیازی به قراردادن علامت + در پشت عدد نیست.
- ۲- در بین ارقام عدد نباید ویرگول قرار دهید تا ارقام عدد، دسته بندی و جدا شوند.
- ۳- اگر عددی را بخواهید در داخل یک متغیر ذخیره کنید که خارج از ظرفیت و گنجایش آن متغیر باشد، مترجم متوجه آن شده و اجازه نمی‌دهد.

مثلاً دستور انتساب زیر را در نظر بگیرید :

```
byte age = 256 ;
```

با توجه به ظرفیت متغیر age که حداکثر عدد ۲۵۵ است، در هنگام ترجمه این دستور، خطای

شکل ۴-۱ ظاهر می‌شود که شرح آن چنین است :

«مقدار ثابت ۲۵۶ را نمی‌تواند به یک byte تبدیل شود».



شکل ۴-۱- خطا در انتساب عدد صحیح ۲۵۶ در یک متغیر نوع byte

در یک برنامه به زبان C# می‌توانید اعداد صحیح را در مبنای ۱۶ نیز بنویسید. برای این منظور قبل از عدد مورد نظر از پیشوند 0x یا 0X استفاده کنید که نشانه اعداد مبنای ۱۶ می‌باشد. مثلاً:

```
byte portValue = 0x1B;
ushort portAddress = 0X00FF;
```

با اجرای این دستورات در متغیر portValue عدد ۲۷ و در متغیر portAddress عدد ۲۵۵ قرار می‌گیرد.

برای مشخص کردن انواع عددی دیگر از نشانه‌های جدول ۲-۴ استفاده می‌شود که در انتهای عدد ذکر می‌شود.

جدول ۲-۴- نشانه‌های نوع اعداد ثابت

نوع عدد	نشانه	مثال
عدد صحیح مثبت unit	U یا u	125U
عدد صحیح بزرگ	L یا l	1700L
عدد صحیح بزرگ مثبت	UL	250000UL
عدد اعشاری با دقت معمولی	F یا f	2.5f
عدد اعشاری با دقت زیاد	D یا d	12.75d
عدد بسیار بزرگ	M یا m	12345678M

نکته

اگر در برنامه، یک عدد اعشاری بدون نشانه بنویسید این عدد به عنوان عدد اعشاری با دقت زیاد در نظر گرفته می‌شود.

برای ذخیره اعداد اعشاری باید از متغیرهای نوع float یا double استفاده کنید. مثلاً برای نگهداری نمرات درسی (معمولاً با دو رقم اعشار) یا اعداد گنگ مانند π باید از چنین متغیرهایی استفاده کرد. دستور زیر را در نظر بگیرید:

```
double PI = 3.141592653589793238;
```

در این دستور برای نگهداری عدد π متغیر PI با دقت زیاد اعلان و مقداردهی شده است.

$$\backslash . (1B) = (1 \times 16) + 11 = 27$$

$$(FF) = (15 \times 16) + 15 = 255$$

نکته

برای ذخیره اغلب داده‌ها مانند نمره یک درس، متغیر نوع float مناسب است. اگر چه می‌توانید از متغیر نوع double نیز استفاده کنید ولی حافظه اشغالی این متغیر دو برابر متغیر نوع float است.

دستورات زیر را در نظر بگیرید :

```
float myPhysicMark;
```

```
myPhysicMark = 17.75f;
```

در دستورات بالا برای ذخیره نمره درس فیزیک متغیری اعلان و مقداردهی شده است.

سؤال؟ در دستور انتساب، بعد از عدد اعشاری 17.75 حرف f نوشته شده است که نشانه اعداد اعشاری با دقت معمولی است. آیا می‌توانید حرف f را بنویسید؟

نکته

در زبان C# هر عدد اعشاری داخل برنامه، به وسیله مترجم به عنوان نوع double در نظر گرفته می‌شود. بنابراین اگر بخواهید یک عدد ممیزی را در یک متغیر نوع float ذخیره کنید مترجم خطا یا هشدار می‌دهد. برای جلوگیری از این مسئله باید از متغیرهای نوع double در هنگام کار با اعداد اعشاری استفاده کنید و یا اینکه در جلوی اعداد اعشاری حرف F یا f را بنویسید تا مترجم، این عدد را به عنوان یک عدد نوع float در نظر بگیرد.

۵-۴- نشان دادن محتوای متغیرها بر روی صفحه نمایش

معمولاً در برنامه‌ها لازم است محتوای متغیرها که شامل داده‌ها و یا نتایج پردازش با اطلاعات بر روی صفحه نشان داده شود تا کاربر از آنها آگاه شود. بدین منظور از متد Write() یا WriteLine() استفاده می‌کنیم که در فصل‌های قبلی برای نمایش یک پیام یا حاصل یک عبارت به کار گرفته شد. مثلاً برای نشان دادن محتوای متغیر age دستور زیر را می‌نویسیم :

```
byte age = 16 ;
```

```
System.Console.WriteLine( age );
```

با توجه به اینکه در متغیر age عدد ۱۶ قرار دارد با اجرای دستور بالا، این عدد روی صفحه کنسول نشان داده می‌شود.

اگر شخص دیگری غیر از شما، این عدد را روی صفحه مشاهده کند، شاید متوجه نشود که این عدد چیست و شاید عدد ۱۶ را به عنوان نمره در نظر بگیرد. بنابراین بهتر است قبل از نمایش هر عدد، یک پیام (رشته) نیز نشان داده شود و به صورت کوتاه و مختصر منظور و مفهوم عددی را که قرار است روی صفحه نشان داده شود بیان کند. بنابراین دستور بالا را به صورت زیر می‌نویسیم :

```
System.Console.WriteLine("My age is " + age);
```

با اجرای این دستور، عبارت زیر روی صفحه نشان داده می‌شود :

```
My age is 16
```

علامت + در دستور بالا، به معنای عمل جمع ریاضی نیست بلکه به منظور کنار هم قرار دادن این دو مقدار (رشته‌ها) استفاده شده است. همان طور که در دستور زیر نیز از علامت + استفاده شده است :

```
System.Console.WriteLine("I am " + age + "years old.");
```

با اجرای این دستور، عبارت زیر روی صفحه نشان داده می‌شود :

```
I am 16 years old.
```

مثال ۱-۴- استفاده از چند متغیر صحیح و اعشاری در برنامه ۱-۴، نشان داده شده است :

```
class VariableDemo
```

```
{
```

```
    static void Main()
```

```
    {
```

```
        // Declare some integer numbers variables
```

```
        int a = 10 , b = 20 , c ;
```

```
        c = a + b;
```

```
        Console.WriteLine("a= " + a);
```

```
        Console.WriteLine("b= " + b);
```

```
        Console.WriteLine("a + b = " + c);
```

```
        // Declare some real numbers variables
```

```
        float lowPI = 3.141592653589793238f;
```

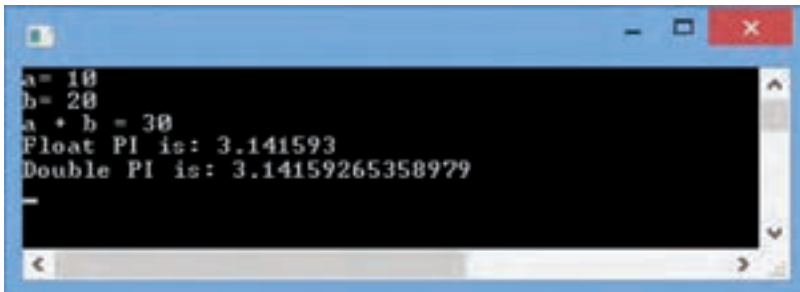
```
        double highPI = 3.141592653589793238;
```

```
// Print the results on the console
Console.WriteLine("Float PI is: " + lowPI);
Console.WriteLine("Double PI is: " + highPI);

Console.ReadKey();
}
}
```

برنامه ۴-۱- تعریف و مقداردهی و نمایش محتوای متغیرها

در برنامه ۴-۱، سه متغیر a , b , c از نوع عدد صحیح تعریف شده‌اند و در متغیر c نتیجه حاصل جمع دو عدد a و b قرار می‌گیرد. در دو متغیر اعشاری $lowPI$ و $highPI$ عدد π با دقت‌های مختلف نگهداری شده است.



```
a = 10
b = 20
a + b = 30
Float PI is: 3.141593
Double PI is: 3.14159265358979
-
```

شکل ۴-۲- خروجی برنامه ۴-۱

۴-۶- نحوه نام گذاری متغیرها

همان طور که پدر و مادر برای انتخاب یک نام خوب و مناسب برای فرزند خود، وقت زیادی می‌گذارند و نکاتی از جمله زیبایی نام و با معنا بودن را رعایت می‌کنند و همچنین سعی می‌کنند که این نام قبلاً در خانواده و یا نزدیکان انتخاب نشده باشد، به همان صورت برنامه نویس نیز برای متغیرها باید یک نام صحیح، با معنا و غیر تکراری در محدوده آن را انتخاب کند این کار باید با حوصله انجام شود و نام انتخابی نباید با نام‌های دیگر یکسان باشد.

در زبان C# در نام گذاری متغیرها، رعایت موارد زیر **الزامی** است :

۱- استفاده از حروف الفبا، اعداد و کاراکتر زیرخط، **مجاز** است.

۲- نام متغیر **نمی تواند** با عدد شروع شود.

۳- نام انتخابی **نمی تواند** با کلمات کلیدی و یا رزرو شده باشد.

۴- استفاده از علامت فاصله و خط تیره در نام متغیر **مجاز نیست**.

در انتخاب نام متغیرها، **بهتر** است نکات زیر رعایت شود :

● نام با معنی و با توجه به کاربرد متغیر در برنامه انتخاب شود. مانند `woodLength`

● از نام های مخفف استفاده نکنید چون خواندن آنها مشکل است. مانند `crntStdnt`

● اولین حرف نام متغیر را با حروف کوچک شروع کنید و اگر نام متغیر از چند کلمه تشکیل شده، برای خوانایی، حرف اول کلمات بعدی را با حروف بزرگ بنویسید. به این روش نوشتن نام، کوهان شتری^۱ می گویند.

چند نمونه نام متغیر دو کلمه ای به روش کوهان شتری را ملاحظه می کنید :

`fileName` , `userName` , `notFound` , `localIP`

روش دیگری برای نام گذاری متغیرها به نام روش مجارستانی^۲ به وسیله آقای چارلز سیمونی^۳ ابداع شده که در ابتدای نام متغیر، مخفف نوع داده ذکر می شود که یک روش شناخته شده و معروف برای نام گذاری متغیرها است.

چند نمونه نام متغیر دو کلمه ای، به روش مجارستانی را ملاحظه می کنید :

`IntNumber` , `LngSalary` , `BlnStatus`

در این کتاب از روش کوهان شتری برای نام گذاری متغیرها استفاده شده است.

نکته

با توجه به حساسیت زبان C# به حروف کوچک و بزرگ، در نام گذاری متغیرها دقت کنید که متغیر `a` و `A` مستقل هستند.

در جدول صفحه بعد تعدادی نام متغیر و علت مجاز یا غیرمجاز بودن این نام ها را می بینید.

۱ _ Camel Notation

۲ _ Hungarian Notation

۳ _ Charles Simonyi

جدول ۳-۴- نمونه متغیرهای مجاز و غیر مجاز

نام متغیر	توضیح
1a	غیر مجاز، نام متغیر نباید با عدد شروع شود.
a1	مجاز
employee Salary	غیر مجاز، بین کلمات نباید فاصله وجود داشته باشد
First	مجاز
Hello!	غیر مجاز، علامت تعجب نباید در نام وجود داشته باشد
payRate	مجاز
one+two	غیر مجاز، علامت + در یک نام نباید قرار داشته باشد
Conversion	مجاز
counter_1	مجاز
2nd	غیر مجاز، نام نمی‌تواند با عدد شروع شود

در دستورات زیر، چند نمونه از اعلان و مقداردهی متغیرها را مشاهده می‌کنید:

```
int speed = 70; // تعریف متغیر برای نگهداری سرعت خودرو با مقداردهی اولیه
float a, b, c; // Triangle sides تعریف سه متغیر برای اضلاع مثلث
float triangleArea; // تعریف یک متغیر برای نگهداری مساحت مثلث
double electricalCharge; // متغیری برای نگهداری بار الکتریکی یک جسم
```

۷-۴- کار با اعداد اعشاری

در فیزیک و شیمی و یا به طور کلی در علوم، با اعداد بسیار کوچک و بسیار بزرگ سروکار داریم. اگر بخواهید عدد اعشاری بسیار کوچک و یا بسیار بزرگی را در یک متغیر ذخیره کنید، می‌توانید آن را به صورت کوتاه با روشی شبیه نماد علمی بنویسید. برای اینکه با این روش آشنا شوید ابتدا لازم است روش نماد علمی را یادآوری کنیم.

در روش نماد علمی، هر عدد از ۲ بخش تشکیل می‌شود که با علامت ضرب از یکدیگر جدا شده‌اند. بخش اول یک عدد اعشاری بین ۱ تا ۹ است (فقط یک رقم صحیح دارد) که به آن مانتیس می‌گویند و قسمت دوم که به صورت توانی از عدد ۱۰ است که به آن نما گفته می‌شود (جدول ۴-۴).

جدول ۴-۴- مثال‌هایی از فرم نماد علمی

فرم معمولی	فرم نماد علمی
4380000	4.38×10^6
.0000265	2.65×10^{-5}
47.9832	4.79832×10^1
10000000	1×10^7
-5600	-5.6×10^3

۱-۷-۴- فرم نقطه شناور: در زبان C# از یک فرم نماد علمی برای نمایش اعداد اعشاری استفاده می‌شود که به آن فرم نقطه شناور^۱ گفته می‌شود. در این فرم مانند نماد علمی، عدد از دو بخش مانتیس و نما تشکیل شده است که با حرف E از یکدیگر جدا شده‌اند. در این فرم، توان 10° بعد از حرف E نوشته می‌شود و خبری از علامت ضرب بین دو قسمت نیست (جدول ۴-۵).

جدول ۴-۵- مثال‌هایی از نمایش اعداد در فرم نقطه شناور

عدد	نمایش عدد در فرم نقطه شناور
75.924	7.5924E1
0.18	1.8E-1
0.0000453	4.53E - 5
-1.482	-1.482E0
7800.0	7.8E3

بار الکتریکی یک الکترون 1.6×10^{-19} کولن است که در متغیرهای زیر ذخیره شده است. می‌توانید این عدد بسیار کوچک را در یک متغیر نوع `double` یا `float` به صورت نقطه شناور ذخیره کنید.

```
double electricalCharge = 1.602E -19;
```

```
float electricalCharges = 1.602E -19F;
```

سؤال: کدامیک از دستورات بالا را ترجیح می‌دهید؟ چرا؟

۲-۷-۴ دقت اعداد قابل نمایش در فرم نقطه شناور : حداکثر تعداد ارقام غیر صفر و با معنی مانتیس عدد را، دقت عدد می‌نامند. دقت اعداد نوع float ۷ رقم و اعداد نوع double ۱۵ رقم است.

نکته

به غیر از میزان حافظه مصرفی و محدوده اعداد قابل نمایش در نوع داده‌های float و double، میزان دقت این دو نوع داده نیز با یکدیگر متفاوت است.

۸-۴- نوع داده منطقی یا بولین^۱ (bool)

در انتهای جدول ۱-۴ نوع داده منطقی یا بولین (bool) را مشاهده می‌کنید این نوع داده فقط شامل دو مقدار درست (true) و نادرست (false) است. متغیرهایی که از این نوع داده تعریف و ایجاد می‌شوند، قادرند یکی از دو مقدار true و false را بپذیرند که با حروف کوچک انگلیسی نوشته می‌شوند. دستورات زیر متغیر response را اعلان و با false مقدار دهی اولیه می‌کند. سپس محتوای متغیر بر روی صفحه نمایش چاپ می‌شود.

```
bool response = false;  
System.Console.WriteLine(response);
```

۹-۴- نوع داده حرفی یا کاراکتری char

کاراکتر عبارت است از یک حرف الفباء یا یک علامت و یا نشانه‌هایی مانند آنچه که در روی دکمه‌های صفحه کلید مشاهده می‌کنید. در کامپیوتر برای هر دکمه صفحه کلید یک کد عددی در نظر گرفته می‌شود و در واقع هنگامی که یک کلید را فشار می‌دهید کدی متناظر با آن کلید تولید و این کد به صورت دنباله‌ای از صفر و یک در حافظه کامپیوتر ذخیره می‌شود. یک کاراکتر را می‌توانید با کد آن مشخص کنید و یا علامت آن را در بین علائم ' ' (تک کوتیشن) قرار دهید. چند نمونه از کاراکترها را در زیر مشاهده می‌کنید.

'A' , 'a' , '&' , '\$' , '+' , ''

^۱ Boolean

نکته

- در داخل علامت‌ها فاصله (Space) نیز به عنوان یک کاراکتر در نظر گرفته می‌شود.
- در داده کاراکتری، فقط یک کاراکتر باید بین علائم ' وجود داشته باشد.

توجه داشته باشید که در زبان برنامه‌نویسی C#، نوع داده char به منظور کار با داده‌های کاراکتری پیش‌بینی شده است. اگر بخواهید یک کاراکتر را در یک متغیر ذخیره کنید باید متغیری از نوع داده char تعریف کنید.

گنجایش این متغیر، دو بایت است و کد کاراکتر را نگهداری می‌کند.

; نام متغیر char

در دستور زیر، متغیری به نام ch از نوع char تعریف و حرف A در آن ذخیره شده است.

```
char ch = 'A';
```

متغیر ch یک متغیر دو بایتی است که در آن کد کاراکتر نگهداری می‌شود. این کد دو بایتی طبق استاندارد یونیکد (Unicode) است. در استاندارد یونیکد، کد هر کاراکتر عددی بین ۰ تا ۶۵۵۳۵ است و تمام نشانه‌ها، علائم و حروف الفباء زبان‌های مختلف کشورها به وسیله این استاندارد کدبندی شده است. این کدبندی مستقل از سیستم عامل، زبان برنامه‌نویسی و سخت افزار است.

در برنامه می‌توانید به جای قرار دادن کاراکتر در علائم ' از کد آنها استفاده کنید، چون کاراکترها فقط محدود به آنچه که بر روی صفحه کلید قرار دارد نیستند. بنابراین با دانستن کد هر کاراکتر می‌توانید آن را در برنامه استفاده کنید. معمولاً برای سادگی، این کد را در مبنای ۱۶ ذکر می‌کنند. با توجه به اینکه در کدبندی یونیکد، از دو بایت استفاده می‌شود و هر ۴ بیت یک رقم مبنای ۱۶ است، برای نمایش این کد در مبنای ۱۶ از یک عدد ۴ رقمی استفاده می‌شود. مثلاً کد کاراکتر A عدد ۶۵ در مبنای ۱۰ است. معادل این کد در مبنای ۱۶ عدد ۴۱ است. این عدد را در داخل علائم ' قرار می‌دهیم و برای مشخص کردن این عدد به عنوان کد کاراکتر، قبل از آن، علامت \u یا \x را می‌نویسیم مانند الگوی زیر:

```
'\u0041'
```

در دستور زیر، متغیر ch اعلان و حرف A در آن ذخیره می‌شود. از صفرهای اضافی قبل از عدد، برای تکمیل کد به صورت ۴ رقمی استفاده شده است.

```
char ch = '\u0041'; // Same as Char ch = 'A'
```

۱۰-۴- نوع داده رشته‌ای (String)

نوع داده char، تنها برای نگهداری یک کاراکتر مناسب است. برای هنگامی که داده‌ها، مانند نام یک شخص، بیش از یک کاراکتر است باید از نوع داده رشته‌ای (string) استفاده کنیم. یک رشته شامل تعدادی حروف و کاراکتر است که در بین جفت کوتیشن " " قرار گرفته است. مثلاً "Mohammad" یک داده رشته‌ای شامل ۸ کاراکتر است.

۱۰-۴- متغیر رشته‌ای: برای نگهداری داده‌های رشته‌ای در برنامه، باید متغیر رشته‌ای تعریف کنید. متغیرهای رشته‌ای قادرند آدرس محلی که یک داده رشته‌ای وجود دارد را نگهداری کنند یا به عبارت ساده، قادرند داده‌های رشته‌ای را ذخیره کنند. بنابراین با متغیری از نوع رشته، قادر خواهیم بود به داده‌های رشته‌ای دسترسی داشته باشیم. دستور زیر یک متغیر رشته‌ای به نام name را اعلان می‌کند.

```
string name;
```

و با دستور انتساب زیر، می‌توانید رشته "Mohammad" را در متغیر name ذخیره کنید و در طول برنامه به آن دسترسی داشته باشید:

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

سؤال: آیا می‌توانید دو دستور بالا را با یک دستور جایگزین کنید؟

۱۰-۴- عملیات بر روی داده‌ها یا متغیرهای رشته‌ای: عملیات مختلفی بر روی رشته‌ها می‌توان انجام داد، یکی از عملیات معمول و کاربردی، الحاق یا کنارهم قرار دادن رشته‌ها است. برای الحاق دو رشته از علامت + استفاده می‌شود.

قطعه کد زیر را در نظر بگیرید. در این کدها، محتوای متغیر رشته‌ای name با رشته "Welcome" الحاق شده و حاصل در متغیر message قرار می‌گیرد.

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

```
string message = "Welcome" + name;
```

```
System.Console.WriteLine(message);
```

نتیجه خروجی چنین خواهد بود:

```
WelcomeMohammad
```

سؤال: اگر بخواهید خروجی به صورت خوانا Welcome Mohammad شود یعنی بین دو کلمه یک فاصله قرار گیرد، چه تغییری در دستورات بالا ایجاد می‌کنید؟

با توجه به این که در زبان C#، علامت + هم برای عمل جمع ریاضی و هم برای الحاق رشته‌ها استفاده می‌شود، در به کارگیری این علامت در برنامه باید دقت کافی داشته باشید.

کود کارگاه ۱

مثال ۲-۴: برنامه زیر مانند برنامه ۱-۴ برای محاسبه مجموع دو عدد a و b نوشته شده است. با این تفاوت که حاصل جمع در متغیری ذخیره نشده بلکه روی صفحه نمایش، نشان داده می‌شود. به خط آخر این برنامه توجه کنید. آیا به نظر شما با اجرای این برنامه، عدد ۲۵ به عنوان حاصل جمع نشان داده می‌شود؟ چرا؟

```
class VariableDemo
{
    static void Main()
    {
        // Declare two integer variables
        int a, b;

        a = 10;
        b = 15;

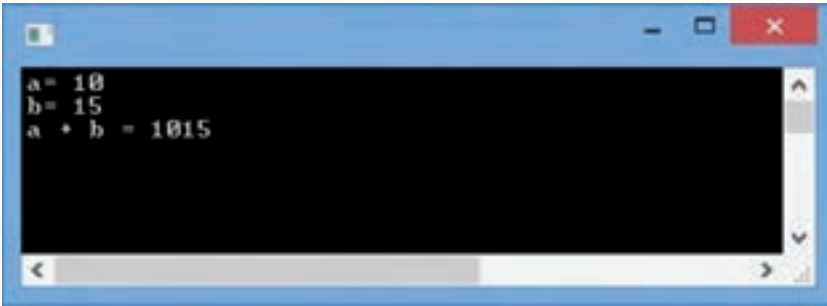
        Console.WriteLine("a= " + a);
        Console.WriteLine("b= " + b);

        // What is displayed?
        Console.WriteLine("a + b = " + a + b); ←
    }
}
```

برنامه ۲-۴ دقت در استفاده از علامت + در هنگام کار با اعداد و رشته‌ها

آشنایی با انواع داده‌ها و متغیرها

در خط آخر برنامه ۴-۲، علامت + دو بار استفاده شده است که هر دو علامت، عمل الحاق رشته را انجام می‌دهند. خروجی این برنامه مطابق شکل ۴-۳ است:



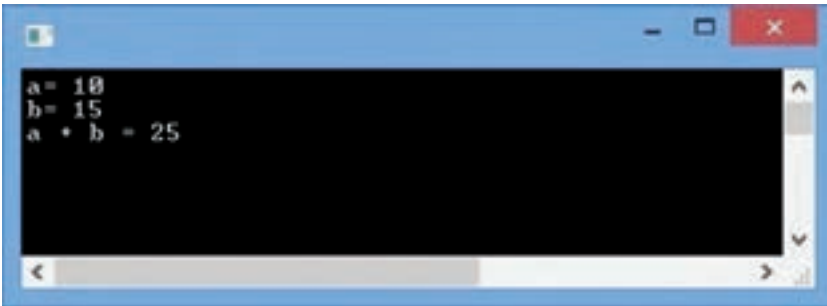
```
a = 10
b = 15
a + b = 1015
```

شکل ۴-۳- خروجی برنامه ۴-۲

برای رفع اشکال برنامه ۴-۲، خط آخر را به صورت زیر بازنویسی می‌کنیم:

```
Console.WriteLine("a + b = " + (a + b));
```

با تصحیح خط آخر، نتیجه اجرای برنامه شکل ۴-۴ خواهد شد:



```
a = 10
b = 15
a + b = 25
```

شکل ۴-۴- خروجی برنامه ۴-۲ پس از تصحیح

سؤال: با توجه به خروجی برنامه ۴-۲ چرا پرانتز سبب تغییر مقدار خروجی شد؟

۱۱-۴ دریافت رشته

تاکنون داده‌های مشخص و ثابتی را در داخل برنامه استفاده کردیم. این داده‌ها به وسیله برنامه‌نویس درون برنامه تعیین شده بود. حال می‌خواهیم برنامه‌های خود را کاربردی کنیم و داده‌ای را از کاربر دریافت کنیم. برای این منظور از متد `ReadLine()` استفاده می‌کنیم که به کاربر اجازه می‌دهد تا داده مورد نظر خود را از طریق صفحه کلید وارد کند.

متد `ReadLine()` مانند متدهایی که تاکنون خوانده‌ایم در کلاس `Console` تعریف شده است و در فضای نامی `System` قرار دارد. بنابراین به صورت زیر استفاده می‌شود:

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

کامپیوتر با اجرای این متد متوقف شده و منتظر دریافت داده می‌شود. کاربر می‌تواند داده مورد نظر خود را تایپ کند و در پایان دکمه `Enter` را بزند که در این صورت، داده به صورت یک رشته در حافظه ذخیره می‌شود. اگر رشته دریافتی را با دستور `انتساب` در یک متغیر رشته‌ای ذخیره کنیم، داده وارد شده، در برنامه قابل دسترسی خواهد بود.

برای مثال می‌خواهیم نام و نام خانوادگی یک شخص را از کاربر سؤال کرده و در برنامه استفاده کنیم. برای این منظور ابتدا دو متغیر رشته‌ای به نام `name` و `family` از نوع رشته‌ای اعلان می‌کنیم و سپس از متد `ReadLine()` برای دریافت نام و نام خانوادگی به صورت زیر استفاده می‌نماییم:

```
string name, family;
```

```
name = System.Console.ReadLine();
```

```
family = System.Console.ReadLine();
```

نکته

متد `ReadLine()` شبیه متد `ReadKey()` است با این تفاوت که متد `ReadKey()` فقط منتظر دریافت یک کلید می‌شود اما در متد `ReadLine()` تا هنگامی که کلید `Enter` زده نشده است کامپیوتر منتظر می‌ماند.

توجه داشته باشید وقتی کامپیوتر منتظر دریافت داده است کاربر باید بداند که چه داده‌ای را لازم است وارد کند (نام، نمره، سن) بنابراین لازم است قبل از استفاده از متد `ReadLine()` یک دستور برای نمایش یک پیام و توضیحی کوتاه در مورد اینکه کامپیوتر منتظر دریافت چه داده‌ای است در برنامه نوشته شود. از متد `Write()` بدین منظور استفاده می‌کنیم.

مثلاً برای دریافت نام کاربر دستورات زیر را می‌نویسیم :

```
string name ;  
System.Console.WriteLine("Enter your name: ");  
name = System.Console.ReadLine();
```

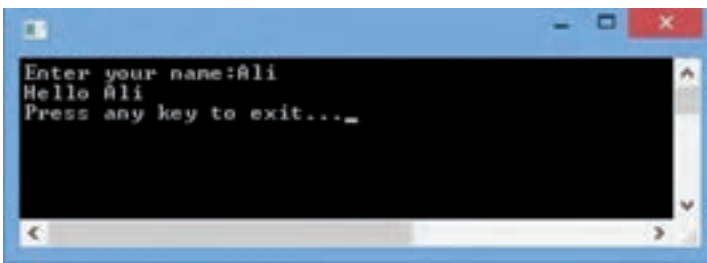
کلودو کارگاه ۲

مثال ۳-۴ : نام کاربر از ورودی دریافت شده و خطاب به او پیام خوشامدگویی اعلام شود.

```
using System;  
class HelloYourName  
{  
    static void Main()  
    {  
        string name;  
        Console.WriteLine("Enter your name: ");  
        name = Console.ReadLine();  
        Console.WriteLine("Hello "+name);  
        Console.WriteLine("Press any key to exit...");  
        Console.ReadKey();  
    }  
}
```

برنامه ۳-۴ خوش آمدگویی به کاربر

اگر فرض کنید که کاربر، نام Ali را وارد کند، خروجی برنامه به صورت شکل ۵-۴ خواهد بود :



شکل ۵-۴ - خروجی برنامه ۳-۴

مثال ۴-۴: می‌خواهیم به برنامه ۳-۴، دستوراتی اضافه کنیم که علاوه بر دریافت نام کاربر، نام خانوادگی وی نیز سؤال شود و سپس نام و نام خانوادگی را در یک خط نمایش دهد. در برنامه ۳-۴، کافی است یک متغیر رشته‌ای به نام family تعریف کرده و از متد ReadLine برای دریافت نام خانوادگی استفاده کنیم. برای نمایش نام و نام خانوادگی در یک خط نیز، از علامت + برای الحاق رشته‌ها استفاده می‌کنیم (کدهای برجسته، تغییرات جدید هستند).

```
using System;
class HelloYourName
{
    static void Main()
    {
        string name, family;
        Console.WriteLine("Enter your name: ");
        name = Console.ReadLine();

        Console.WriteLine("Enter your family: ");
        family = Console.ReadLine();

        Console.WriteLine("Hello " + name + " " + family);

        Console.WriteLine("Press any key to exit. . .");
        Console.ReadKey();
    }
}
```

برنامه ۴-۴ تکمیل برنامه خوش آمدگویی به کاربر

سؤال! خروجی برنامه تغییر یافته چه تفاوتی با شکل ۴-۵ دارد؟

مثال ۴-۵: می‌خواهیم برنامه‌ای بنویسیم که دو عدد دلخواه از کاربر دریافت کند و مجموع آن‌ها را حساب کرده و روی صفحه نمایش، نشان دهد.

برای دریافت داده‌ها از کاربر، از متد `ReadLine()` مانند مثال‌های قبلی استفاده می‌کنیم. داده‌های دریافتی به وسیلهٔ این متد، در قالب رشته در حافظه ذخیره می‌شوند، بنابراین برای دسترسی به آن‌ها باید از متغیرهای رشته‌ای استفاده کنیم.

```
using System ;
class GetNumbers
{
    static void Main()
    {
        string firstNumber, secondNumber;

        Console.WriteLine("Enter a number: ");
        firstNumber = Console.ReadLine();

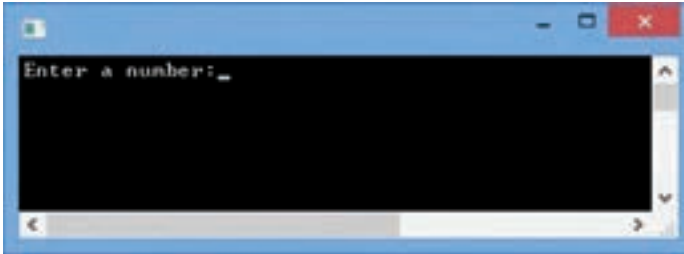
        Console.WriteLine("Enter another number: ");
        secondNumber = Console.ReadLine();

        Console.WriteLine("Total=" + (firstNumber + secondNumber) );

        Console.WriteLine("Press any key to exit...");
        Console.ReadKey();
    }
}
```

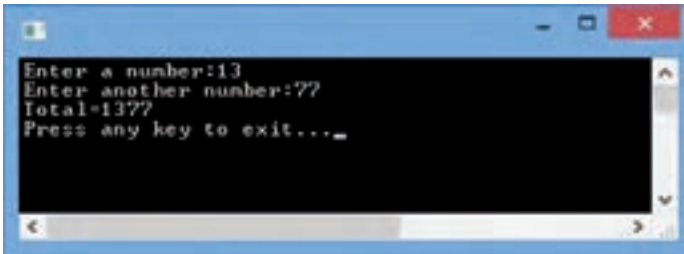
برنامه ۴-۵- اولین تلاش برای دریافت داده‌های عددی

با اجرای این برنامه، پنجره‌ای ظاهر می‌شود که از کاربر خواسته می‌شود که یک عدد وارد کند (شکل ۴-۶).



شکل ۴-۶- خروجی برنامه ۴-۵ دریافت یک عدد

پس از وارد کردن یک عدد و زدن دکمه Enter، عدد دیگری خواسته می‌شود. فرض کنید اعداد ۱۳ و ۷۷ توسط کاربر وارد شود (شکل ۴-۶).



شکل ۴-۷- خروجی برنامه ۴-۵

سؤال! کاربر با وارد کردن دو عدد ۱۳ و ۷۷ انتظار داشت که مجموع آنها یعنی عدد ۹۰ روی صفحه نشان داده شود اما به جای آن، عدد ۱۳۷۷ نشان داده شد. چرا؟

همان طور که بیان شد متد `ReadLine()` داده دریافتی را به صورت یک رشته در حافظه ذخیره می‌کند و در برنامه ۴-۵ از متغیرهای رشته‌ای `firstNumber` و `secondNumber` برای دسترسی به داده‌های ورودی استفاده کردیم. بنابراین علامت + در دستور زیر عمل الحاق دو رشته مثلاً "13" و "77" را انجام می‌دهد و طبیعی است که نباید انتظار عمل جمع ریاضی داشته باشیم.

۱-۱-۴ دریافت اعداد : با توجه به این که داده‌های دریافتی به وسیله متد `ReadLine()` همواره به صورت رشته تحویل داده می‌شود باید به وسیله دستوری، رشته دریافتی را به عدد تبدیل کنیم. بنابراین به متدی نیاز داریم که بتواند یک رشته شامل ارقام را به ارزش عددی تبدیل کند تا بتوانیم روی آنها محاسبات ریاضی انجام دهیم.

خوشبختانه برای انواع داده‌های عددی، متدی به نام `Parse()` از قبل تعریف شده است که می‌تواند از یک رشته شامل ارقام، معادل عددی آن را بدست آورد. مثلاً برای تجزیه رشته "259" به ارزش عددی، با توجه به این که درون رشته، یک عدد صحیح قرار دارد، از متد `Parse()` مربوط به نوع داده `int` استفاده می‌کنیم:

```
int.Parse("259");
```

نکته

به عمل بررسی کاراکتر به کاراکتر یک رشته، برای جدا کردن و بدست آوردن یک مقدار با معنی، تجزیه کردن^۱ می‌گویند.

حاصل اجرای این متد، عدد ۲۵۹ است که باید در یک متغیر نوع صحیح ذخیره شود. بنابراین استفاده مفید از این متد به صورت زیر خواهد بود:

```
int a ;
```

```
a = int.Parse("259");
```

می‌توانید دو دستور بالا را با دستور زیر جایگزین نمایید:

```
int a = int.Parse("259");
```

اگر رشته‌ای حاوی عدد اعشاری باشد باید از متد `Parse()` مربوط به نوع داده اعشاری مثلاً `float` یا `double` استفاده کنید. مثلاً برای تبدیل رشته "2.50" به عدد 2.5 از دستورات زیر استفاده می‌کنیم:

```
float b ;
```

```
b = float.Parse("2.50");
```

با استفاده از متد `Parse()` می‌توانیم رشته دریافتی که به وسیله متد `ReadLine()` از کاربر گرفته شده است را به عدد تبدیل کنیم به شرط اینکه حاوی اعداد باشد.

```
string input;
```

```
float number;
```

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

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

^۱_Parse

همچنین می‌توانید متد `ReadLine()` را مستقیماً در متد `Parse()` استفاده کنید که در این صورت نیازی به متغیر رشته‌ای نیست :

```
float number;
number = float.Parse(Console.ReadLine());
```

سؤال: آیا می‌توانید دو دستور بالا را، باز هم خلاصه‌تر کنید؟

کاربرد کارگاه ۳

مثال ۴-۶ : با تکمیل برنامه ۴-۵ مجموع دو عدد دریافتی را چاپ نمایید.

```
using System ;
class GetNumbers
{
    static void Main()
    {
        string input;
        float firstNumber, secondNumber;

        Console.WriteLine("Enter a number: ");
        input = Console.ReadLine();
        firstNumber = float.Parse(input);

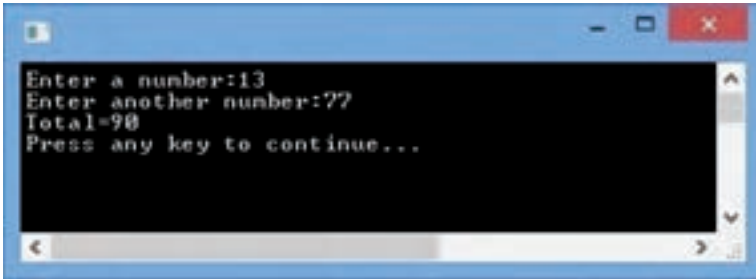
        Console.WriteLine("Enter another number: ");
        input = Console.ReadLine();
        secondNumber = float.Parse(input);

        Console.WriteLine("Total=" + (firstNumber + secondNumber));

        Console.WriteLine("Press any key to continue. . .");
        Console.ReadKey();
    }
}
```

برنامه ۴-۶- دریافت دو عدد و محاسبه مجموع

نتیجه اجرای برنامه در شکل ۴-۸ نشان داده شده است :



شکل ۴-۸- خروجی برنامه ۴-۶

برنامه‌های زیر را در محیط VS ایجاد کنید.

۱- دستورات زیر را در داخل متد Main() برای شناسایی انواع متغیرها بنویسید. با اضافه کردن دستورات WriteLine()، محتوای متغیرها را بر روی صفحه نمایش، نشان دهید.

```
// Declare and initialize some variables
```

```
// Use long suffix.
```

```
long aLongNumber = 10000L;
```

```
// Use double suffix.
```

```
double aDoubleNumber = 123.764D;
```

```
// Use float suffix.
```

```
float aFloatNumber = 100.50F;
```

```
// Use unsigned suffix.
```

```
uint anUnsignedNumber = 1000U;
```

```
// Use decimal suffix.
```

```
decimal aDecimalNumber = 4000.1234M;
```

```
// Use unsigned suffix and long suffix.
```

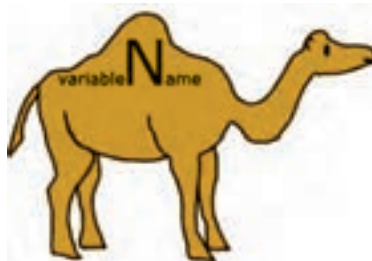
```
ulong anUnsignedLong = 10002000300040005000UL;
```

۲- برنامه شماره ۴-۴ (خوش آمدگویی به کاربر) را با داده‌های مختلف (نام خود، نام همکلاسی‌ها) آزمایش کنید.

۳- برنامه شماره ۴-۶ (دریافت دو عدد و محاسبه مجموع) را با اعداد صحیح و اعشاری آزمایش کنید.

خودآزمایی فصل چهارم

- ۱- چه نوع حافظه کامپیوتر برای نگهداری حجم کمی از داده‌ها در طول اجرای یک برنامه مناسب است؟
- ۲- در زبان‌های برنامه‌نویسی، مکانی از حافظه برای نگهداری موقتی داده و اطلاعات نامیده می‌شود.
- ۳- منظور از نوع داده چیست؟
- ۴- نوع متغیر چه ویژگی‌هایی را نشان می‌دهد؟
- ۵- تفاوت‌های بین نوع داده float و double را نام ببرید.
- ۶- برای نگهداری هر یک از داده‌های زیر در برنامه، یک متغیر مناسب تعریف کنید.
 - الف) سن افراد
 - ب) درجه حرارت محیط اتاق
 - ج) وضعیت خاموش و روشن بودن یک لامپ
 - ث) جمعیت یک کشور
- ۷- تحقیقی کوتاه بر روی روش نام گذاری مجارستانی (Hungarian Notation) با استفاده از اینترنت داشته باشید. با توجه به محیط‌های برنامه‌نویسی (IDE) پیشرفته، مانند ویژوال استودیو، نشان دهید که دیگر نیازی به ذکر نوع داده در ابتدای نام متغیر که در روش مجارستانی استفاده می‌شود، وجود ندارد.
- ۸- شکل زیر چه روشی را برای نام گذاری متغیرها نشان می‌دهد؟ نام این روش چیست؟



- ۹- چرا در هنگام ترجمه دستور زیر، خطا ظاهر می‌شود؟ چگونه این خطا را برطرف می‌کنید؟

`short value = 66000 / 2 ;`

۱۰- در جدول زیر، کدامیک از نام‌های متغیر، غیر مجاز است و یا مناسب نیستند. آنها را تصحیح کنید. (اولین ردیف جدول برای شما پاسخ داده شده است.)

نام متغیر	کاربرد متغیر با توجه به معنی آن	نام متغیر			نام پیشنهادی شما
		مجاز	نامناسب	غیرمجاز	
networkOK	وضعیت شبکه	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
29yesitsme		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
myCurrentID		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
intValue		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
8%/tax		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
woodLength		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
glassArea		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
width		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
height		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
MySalary		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
employeeSalary		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

۱۱- کدامیک از داده‌های زیر یک داده کارا کتری محسوب می‌شود؟ برای پاسخ خود دلیل بیاورید.

'! ', 'abc', '+', '!', '&', '\', '\۲', '\۰۸'

۱۲- در زبان C#, برای دریافت داده از ورودی، از متد و برای

نمایش اطلاعات در خروجی، از متد استفاده می‌کنیم.

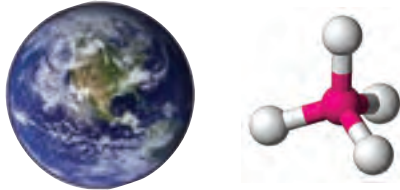
آشنایی با انواع داده‌ها و متغیرها

۱۳- اعداد زیر را در متغیرهای مناسب با استفاده از روش نقطه شناور جای دهید.

الف) عمر زمین بر حسب سال $4,600,000,000$

ب) فاصله زمین تا خورشید $149,600,000,000$ متر است.

پ) اندازه جرم یک اتم کربن (جرم اتمی) برابر با 166kg 10^{-27} است.



۱۴- با توجه به اینکه لازم نیست ماتریس بین 0° تا 1° باشد، جدول زیر را تکمیل کنید.

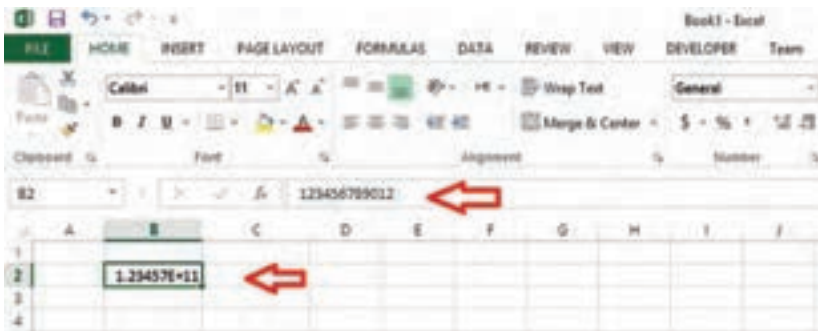
عدد	نمایش عدد در فرم نقطه شناور	گونه دیگری از فرم نقطه شناور
75.924	7.5924E1	
0.18	1.8E-1	
0.0000453	4.53E-5	
-1.482	-1.482E0	
7800.0	7.8E3	

۱۵- در برنامه‌های صفحه گسترده، اعداد بزرگ به صورت نقطه شناور نیز نمایش داده می‌شوند. وارد

برنامه اکسل شوید و عدد بزرگی را وارد نموده و کلید Enter را بزنید. چه عددی روی صفحه نشان داده می‌شود؟

پس از مشاهده فرم نقطه شناور، سعی کنید با تغییر فرمت سلول مربوطه، عدد را به صورت

معمولی نشان دهید.



۱۶- کدام یک از دستورات زیر، می‌تواند مجموع دو عدد a و b را به‌طور صحیح نشان دهد؟ نتیجه اجرای هر یک از دستورات را نیز بنویسید.

```
int a=10,b=20;
System.Console.WriteLine("a"+"b");
System.Console.WriteLine("a+b");
System.Console.WriteLine(a+b);
System.Console.WriteLine("a+b"+a+b);
System.Console.WriteLine("a+b"+(a+b));
```

۱۷- سؤال زیر به زبان انگلیسی است. آن را خوانده و پاسخ صحیح را انتخاب کنید.

The C# method that prints a line of output on the screen and then positions the cursor on the next line is

- | | |
|------------------|----------------|
| A) println() | C) WriteLine() |
| B) DisplayLine() | D) Write() |

تمرینات برنامه‌نویسی فصل چهارم

- برنامه‌ای بنویسید که دو عدد از کاربر دریافت نماید و حاصل جمع و حاصل تفریق آنها را نمایش دهد. (اعداد ورودی ممکن است صحیح و یا اعشاری باشد).
- برنامه‌ای بنویسید که نام و نام خانوادگی و سن کاربر را دریافت کند و سپس اطلاعات دریافتی را با رنگ‌های دلخواه روی صفحه نمایش، نشان دهد.
- با استفاده از یک برنامه ساده شامل متد `WriteLine()`، رشته‌ها یا کاراکترهای جدول زیر را نمایش دهید تا بتوانید معادل آن‌ها را پیدا کرده و جدول را کامل کنید.

معادل	کد حرف یا رشته	معادل	کد حرف یا رشته
	'\u0007'		"0\u00200"
	"b\u0061ck"		"C\u0023"
	'\u000a'		'\u0040'
	"12" + "8"		'\u0030'

واژگان و اصطلاحات انگلیسی فصل چهارم

ردیف	واژه انگلیسی	معنی واژه به فارسی
۱	Assignment	
۲	Boolean	
۳	Built-In Data Type	
۴	Camel Notation	
۵	Concatenate	
۶	Floating point notation	
۷	Hungarian Notation	
۸	Initialize	
۹	Integer Numbers	
۱۰	Precision	
۱۱	Primitive Data Type	
۱۲	Random Access Memory	
۱۳	Representation	
۱۴	Significant digits	
۱۵	Unsigned numbers	
۱۶	Variable	