زنگ سی‌شارپ – قسمت چهارم

نوشته‌ی مسعود درویشیان

لینک مستقیم این مطلب در وب‌تارگت

در قسمت سوم توضیحات مختصری در مورد value type، متغیر (variable) و عملگرها (operators) داده شد در این قسمت با چند مثال به تشریح کامل‌تر آن‌ها خواهیم پرداخت.

همان‌طور که در قسمت قبل گفته شد، data types به دو دسته‌ی value types و reference types تقسیم می‌شوند و دانستید که سیزده value types داریم. در مجموع به این سیزده ولیو تایپ، simple types می‌گویند و دلیل این نام‌گذاری این است که این‌ها شامل مقدار تکی (single value) هستند و به‌عبارت دیگر، ترکیبی از دو یا بیشتر از دو مقدار نیستند.

Float و Double

به مثال زیر توجه کنید:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ExFloat
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 25;
            float f = 16.8F;
            float result = i / f;

            Console.WriteLine("Result is: " + result);
        }
    }
}
```

همان‌طور که مشاهده می‌کنید متغیر i مقدار ۲۵ را در خود ذخیره کرده است و متغیر f که از جنس float است مقدار ۱۶/۸ را در خود نگه‌داری می‌کند. نکته‌ی float این‌جاست که باید بعد از آن از پسوند F یا f استفاده کنید:

```
float f = 16.8F;
```

دلیل این کار این است که اعداد اعشاری به صورت پیش‌فرض **double** هستند و برای این‌که یک مقدار را به‌صورت **float** ذخیره کنید باید حتماً از پسوند **F** یا **f** استفاده کنید، در غیر این صورت کامپایلر به شما پیغام خطا می‌دهد و به شما می‌گوید که نمی‌توانید یک مقدار از جنس **double** را در متغیری از جنس **float** ذخیره کنید.

```
float result = i / f;
```

در این‌جا مقدار متغیر **i** را بر مقدار متغیر **f** تقسیم کردیم و نتیجه‌ی آن‌را در متغیری به‌اسم **result** که از جنس **float** است ذخیره کردیم.

```
Console.WriteLine("Result is: " + result);
```

در نهایت مقدار **result** را در خروجی نمایش دادیم:

```
Result is: 1.488095
```

اگر همین مثال را برای **double** انجام دهیم همان‌طور که در قسمت قبل گفته شد خواهیم دید که دقت دابل بیشتر از **float** است:

```
static void Main(string[] args)
{
    int i = 25;
    double d = 16.8;
    double result = i / d;

    Console.WriteLine("Result is: " + result);
}
```

خروجی:

```
Result is: 1.48809523809254
```

Decimal

یکی از چیزهای خیلی عالی در مورد سی‌شارپ تدارک دیدن نوع **Decimal** برای محاسبات مالی است. نوع دسیمال با بهره‌گیری از ۱۲۸ بیت برای نشان دادن مقادیر در محدوده‌ی ۱×۱۰^{-۲۸} و $۷/۹ \times ۱۰^{۲۸}$ استفاده می‌کند. در محاسبات معمولی ممیز شناور، خطاهای گرد کردن گوناگونی رخ می‌دهد. نوع دسیمال این خطاها را از بین می‌برد و دقیقاً تا ۲۸ رقم اعشار (در بعضی موارد ۲۹ رقم اعشار) را نشان می‌دهد. این توانایی نشان دادن مقادیر اعشاری بدون خطای گرد کردن، نوع **Decimal** را برای محاسبات پولی و مالی بسیار مناسب می‌کند.

به‌دلیل این‌که مقادیر پولی در اپلیکیشن‌های تجاری بسیار با اهمیت هستند، بسیاری از برنامه‌نویسان سی‌شارپ اغلب برای این منظور از نوع **Decimal** استفاده می‌کنند. برای مشخص کردن این‌که عدد شما از نوع **Decimal** است، باید به عدد خود کاراکتر **M** (یا **m**) را اضافه کنید:

```
decimal d = 12.30M;
```

این‌کار ضروری است چراکه در غیر این‌صورت مقدار متغیر به‌عنوان **double** تفسیر می‌شود (مقادیر اعشاری به‌صورت پیش‌فرض **double** هستند).

در این‌جا یک مثال برای شما در نظر گرفته‌ایم که با استفاده از نوع **Decimal** چگونه قیمت نهایی همراه با تخفیف را با توجه به قیمت اصلی و درصد تخفیف به‌دست آورید:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace test
{
    class Program
    {
        static void Main(string[] args)
        {
            decimal price;
            decimal discount;
            decimal discountedPrice;

            // Compute discounted price.
            price = 19.95m;
            discount = 0.15m; // discount rate is 15%

            discountedPrice = price - (price * discount);

            Console.WriteLine("Discounted price: $" + discountedPrice);
        }
    }
}
```

خروجی این برنامه:

```
Discounted price: $16.9575
```

در این برنامه، در قسمتی که محاسبه تخفیف انجام می‌شود، الویت اول با پرانتزها است به‌طوری که ابتدا متغیر **price** در **discount** ضرب می‌شود، سپس حاصل آن از متغیر **price** کم می‌شود و در نهایت مقدار محاسبه شده‌ی نهایی در متغیر **discountedPrice** قرار می‌گیرد.

دریافت ورودی از کاربر

به مثال زیر که برای دریافت مقدار از ورودی است توجه کنید:

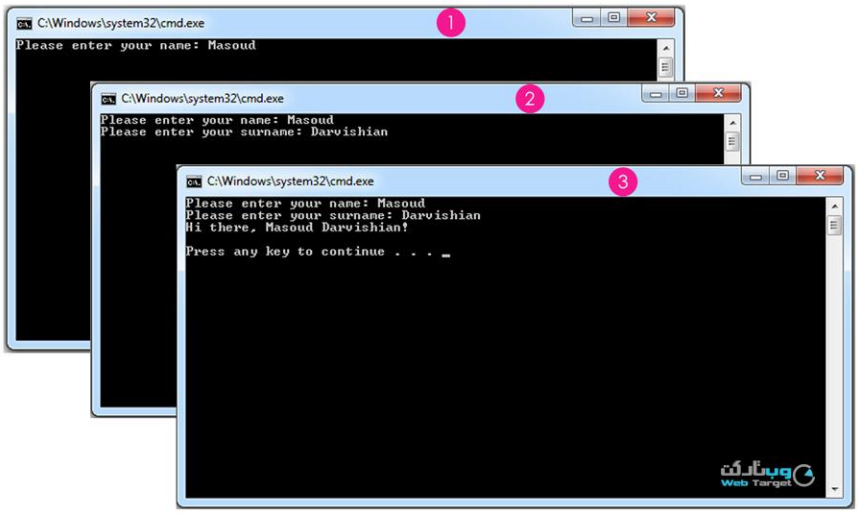
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace test
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Please enter your name: ");
            string userName = Console.ReadLine();

            Console.Write("Please enter your surname: ");
            string userSurname = Console.ReadLine();

            Console.WriteLine("Hi there, {0} {1}!", userName, userSurname);
            Console.WriteLine();
        }
    }
}
```

و خروجی با توجه به اسم و فامیل شما این‌چنین است:



در این مثال ما نام و نام‌خانوادگی کاربر را دریافت می‌کنیم و در نهایت یک پیغام خوش‌آمدگویی به کاربر نمایش می‌دهیم. هدف از این مثال این است که شما بیاموزید چگونه یک مقدار (ورودی) را از کاربر دریافت کنید و چگونه آن‌را نمایش دهید و در صورت نیاز چه تغییراتی روی آن اعمال کنید.

به توضیحات برنامه بالا توجه کنید:

```
Console.Write("Please enter your name: ");
```

با این خط‌کد شما به کاربر یک پیغام را نمایش می‌دهید که نام خودش را وارد کند. خب کاربر نام خودش را وارد می‌کند ولی این نام وارد شده که مسلماً **string** (رشته‌ای از کاراکترها) است در کجا ذخیره می‌شود؟ پس شما باید یک جا را برای گرفتن این مقدار آماده کنید. کامپیوتر در نهایت همه‌ی اطلاعات را در **خانه‌های حافظه (Ram)** ثبت می‌کند که به صورت **باینری** هستند. حتی برنامه‌نویسان کارکشته و باتجربه هم ترجیح می‌دهند به‌جای استفاده از باینری، مستقیماً از اعداد، متن و نوشته استفاده کنند. زبان‌های برنامه‌نویسی به شما اجازه می‌دهند که مکان‌های خاصی را برای نگه‌داری اطلاعات در حافظه به‌وجود بیاورید، این مکان‌های خاص همان متغیرها هستند که در قسمت قبل به شما معرفی کردیم.

شما تا این‌جا یک پیغام را به کاربر نمایش دادید که نام خودش را وارد کند و اکنون می‌خواهیم نام کاربر را دریافت و ذخیره کنیم، خط کد بعدی این کار را برای ما انجام می‌دهد:

```
string userName = Console.ReadLine();
```

در این‌جا برای دریافت نام کاربر یک متغیر به اسم **userName** از جنس **string** تعریف کردیم تا نام کاربر را در آن ذخیره کنیم. اکنون ما نیاز داریم که این مقدار را از کاربر دریافت کنیم. دستور **Console.ReadLine()** این کار را برای ما انجام می‌دهد و به‌این صورت عمل می‌کند که منتظر می‌ماند تا کاربر مقداری را در پنجره کنسول تایپ کند و به‌محض این‌که کاربر کلید **Enter** را فشرد، هرچه را که تایپ کرده در متغیر **userName** ذخیره می‌کند. در این‌جا ما هم‌زمان هم متغیر را تعریف کردیم هم مقدار ورودی را در آن قرار دادیم، ولی می‌توانستیم ابتدا متغیر را تعریف کنیم، سپس مقدار ورودی را در آن قرار دهیم، بدین صورت:

```
string userName;
userName = Console.ReadLine();
```

برای دریافت نام‌خانوادگی هم طبق همین روال پیش می‌رویم. همین‌طور که می‌بینید ما برای انتخاب نام متغیرها از **userName** برای اسم کاربر و از **userSurname** برای نام‌خانوادگی کاربر استفاده کردیم. یک‌سری قرارداد و راهنمایی برای نام‌گذاری وجود دارد که به شما کمک می‌کند چگونه یک نام بسیار مناسب را انتخاب کنید تا اگر در زمان آینده به برنامه‌ی خود برگشتید دچار سردرگمی نشوید و علاوه بر آن از یک خوش‌نویسی در زبان سی‌شارپ بهره ببرید، به چند نمونه از این قراردادها و راهنمایی‌ها توجه کنید:

- از space و نقطه‌گذاری نمی‌توانید در انتخاب نام استفاده کنید.
- از نام‌های ساده، خوانا و بامعنی استفاده کنید.
- خوانایی و بامعنی بودن یک اسم را قربانی یک اسم کوتاه و عجیب و غریب نکنید.
- از زیرخط، خط فاصله یا هر کاراکتری غیر از کارکترهای الفبایی استفاده نکنید.
- از انتخاب اسم‌هایی که با کلمات کلیدی زبان برنامه نویسی تداخل دارند پرهیز کنید.
- سی‌شارپ یک زبان **case-sensitive** است و اکثر برنامه‌نویسان به‌طور عمده از حروف کوچک برای نام‌گذاری متغیرها استفاده می‌کنند ولی اگر نام متغیر چندقسمتی باشد برای تمایز بین کلمات از حروف بزرگ استفاده می‌کنند برای مثال **userName** و **userSurname** که برای نام‌گذاری آن‌ها را انتخاب کردیم، دو قسمتی (دو کلمه‌ای) هستند و **userName** از دو کلمه‌ی **user** و **name** تشکیل شده است که برای تمایز بین آن‌ها حرف **N** را بزرگ نوشته‌ایم و هرکلمه‌ای که به‌این اسم به‌خواهد اضافه شود حرف اول آن باید بزرگ باشد.

```
Console.WriteLine("Hi there, {0} {1}!", userName, userSurname);
```

اگر شما این خط‌کد را با خروجی مقایسه کنید متوجه خواهید شد که چه اتفاقی افتاده است. رشته‌ی **"Hi there, "** در خروجی نمایش داده می‌شود و نام کاربر به‌جای **{۰}** قرار می‌گیرد و نام‌خانوادگی کاربر در قسمت **{1}** واقع می‌شود. اگر شما به‌خواهید مقدار یک متغیر را در خروجی همراه یک پیغام نمایش دهید می‌توانید متغیر را با یک عدد در کروشه جایگزین کنید. کامپیوتر شمارش را از عدد صفر شروع می‌کند بنابراین **userName** متغیر شماره صفر می‌شود و مقدار آن در خروجی چاپ می‌شود همچنین **userSurname** متغیر شماره یک است. همان‌طور که می‌بینید بعد از **string** نام متغیرهایی را که می‌خواهیم مقدار آن‌ها همراه با پیغام خوش‌آمدگویی نمایش داده شود، می‌نویسیم و به‌ترتیب **username** به‌جای **{۰}** و **userSurname** به‌جای **{۱}** قرار گرفته می‌شود. اگر توجه کرده باشید همین کار را با استفاده از علامت **+** هم می‌توانستیم انجام دهیم که بدین صورت می‌شد:

```
Console.WriteLine("Hi there, " + userName + " " + userSurname + "!" );
```

همان‌طور که می‌بینید توسط علامت **+** متغیرها را به رشته‌ها متصل کردیم.

دستور آشنای آخر:

```
Console.WriteLine();
```

این دستور وقتی به‌این صورت و بدون هیچ ورودی استفاده شود تنها یک خط خالی را چاپ می‌کند و معمولاً برای این‌که خروجی‌ها خواناتر باشند استفاده می‌شود.

کلیه حقوق مادی و معنوی برای وب‌سایت وب‌تارگت محفوظ است.
استفاده از این مطلب در سایر وب‌سایت‌ها و نشریات چاپی تنها با ذکر و درج لینک منبع مجاز است.