زنگ سی‌شارپ – قسمت شانزدهم

نوشته‌ی مسعود درویشیان

لینک مستقیم این مطلب در وب‌تارگت

در برنامه‌نویسی بارها مواردی به‌وجود می‌آید که یک نوع از یک متغیر را به نوع دیگری اختصاص دهید. به‌عنوان مثال گاهی پیش می‌آید که مقدار یک int را به یک float اختصاص دهید. یا در مواردی که عملیات ریاضی انجام می‌دهید، نتیجه‌ی محاسبات از جنس متغیر شما خواهد بود. برای مثال هنگامی که دو int را بر هم تقسیم می‌کنید، نتیجه از جنس int خواهد بود.

```
int i;
double b;

i = 180;
b = i; // assing an int to a double
```

هنگامی که دو نوع سازگار با هم ترکیب می‌شوند، مقدار سمت راست به‌صورت اتوماتیک به نوع متغیر سمت چپ کانورت می‌شود. بنابراین در برنامه بالا، مقدار i ابتدا به double کانورت (تبدیل) شده و سپس به b اختصاص می‌یابد.

به‌علت سخت‌گیری سی‌شارپ در بررسی نوع داده‌ها، هر نوع داده‌ای برای تبدیل شدن به نوع دیگر سازگار نیست و به‌صورت اتوماتیک کانورت نمی‌شود که در این موارد برای کانورت می‌توان از cast کردن استفاده کرد.

برای تبدیل نوع داده به نوع دیگر و کانورت کردن، دو روش موجود است:

- Implicit
- Explicit

در روش implicit تبدیل نوع به‌صورت اتوماتیک اتفاق می‌افتد به‌شرطی که:

۱. هر دو نوع داده با هم سازگاری داشته باشند.
۲. بازه و محدوده‌ی نوع مقصد بزرگ‌تر از بازه و محدوده‌ی نوع مبدا باشد.

هنگامی که این دو شرط برقرار باشند تبدیل نوع به‌صورت اتوماتیک (implicit) انجام می‌شود و به اصطلاح یک widening conversion اتفاق می‌افتد. به‌عنوان مثال نوع int محدوده و بازه‌ی کافی برای نگه‌داری نوع byte را در خود دارد. همچنین int و byte دو نوع سازگار هستند. بنابراین یک کانورت به روش implicit می‌تواند اتفاق افتد.

به مثال زیر توجه کنید:

```
using System;
class Example
{
    static void Main()
    {
        int i = 300;
        byte b = 255;

        /* It's a implicit conversion */
        i = b; // assing a byte to an int

        Console.WriteLine("We assing a byte to an int: " + i);
    }
}
```

در این مثال یک implicit conversion اتفاق می‌افتد زیرا هر دو شرط برای این تبدیل نوع برقرار است.

اگرچه برای اختصاص دادن byte به int کانورت به‌صورت implicit اتفاق می‌فتد اما برای اختصاص دادن int به byte این اتفاق نمی‌افتد و به اصطلاح widening conversion ای در کار نیست زیرا یکی از آن دو شرط برای implicit conversion نقض شده است (بازه و محدوده‌ی نوع مقصد باید بزرگ‌تر از بازه و محدوده‌ی نوع مبدا باشد).

به مثال زیر که نادرست است و اجرا نمی‌شود توجه کنید:

```
using System;
class Example
{
    static void Main()
    {
        /* This program will not compile */

        int i = 150;
        byte b;

        b = i; // Illegal!!!

        Console.WriteLine(b);
    }
}
```

البته با یک تغییر کوچک برنامه بالا قابل اجرا خواهد بود که در ادامه به شرح آن می‌پردازیم. اگر توجه کرده باشید هنگام اجرای برنامه بالا این پیغام خطا را دریافت می‌کنید:

Connot implicitly convert type ‘int’ to ‘byte’. An explicit conversion exists (are you missing a cast?)

برای نوع داده‌هایی که با هم سازگاری ندارند و به‌صورت implicit نمی‌توانند کانورت شوند باید از از روش explicit و cast کردن استفاده کرد. در cast کردن ما به کامپایلر دستور می‌دهیم که نوع یک متغیر را آن‌طور و به آن‌چه که می‌خواهیم تبدیل کند (روش explicit).

فرم کلی cast کردن به شکل زیر است:

```
(target-type) expression
```

در این‌جا، *target-type* مشخص کننده‌ی نوعی است که شما خواسته‌اید *expression* به آن تبدیل شود.

به‌عنوان مثال:

```
double x, y;
```

اگر شما بخواهید که حاصل x / y از جنس int باشد می‌توانید بنویسید:

```
(int) (x / y);
```

همان‌طور که می‌دانید x و y از جنس double هستند اما از طریق cast کردن حاصل آن‌ها تبدیل به int شده است.

به این مثال توجه کنید:

```
using System;
class Example
{
    static void Main()
    {
        double x, y;

        x = 25.3;
        y = 5.1;

        int result = (int) (x / y);
        Console.WriteLine("The result is " + result);
    }
}
```

در برنامه بالا x و y که هردو از جنس double هستند بر هم تقسیم شده‌اند. حاصل این تقسیم مسلماً از جنس double خواهد بود اما ما از طریق cast کردن آن را به int تبدیل کرده‌ایم. پرانتزهای اطراف x و y ضروری هستند زیرا در صورت نبود آن‌ها تنها x به int تبدیل می‌شود.

در cast کردن اگر نوع متغیر مقصد کوچک‌تر از مبدا باشد ممکن است بخشی از اطلاعات از بین برود. همچنین هنگام cast کردن مقادیر اعشاری به عدد صحیح قسمت اعشاری آن‌ها از بین می‌رود. برای مثال هنگام cast کردن مقدار ۱.۲۳

به عدد صحیح نتیجه ۱ خواهد بود. همچنین هنگامی که قصد دارید نوع int را در نوع byte قرار دهید مقداری از اطلاعات ممکن از از بین برود چراکه نهایت مقداری که byte می‌تواند ذخیره کند عدد ۲۵۵ است:

```
using System;
class Example
{
    static void Main()
    {
        int anOkayInt = 345;
        byte aBadByte = (byte)anOkayInt;

        Console.WriteLine(aBadByte);
    }
}
```

خروجی این برنامه عدد ۸۹ است. اگر قصد داشته باشید عدد ۲۵۶ را از طریق cast کردن در byte ذخیره کنید، عددی که ذخیره می‌شود صفر است. اگر ۲۵۷ را در byte ذخیره کنید، عدد ۱ ذخیره می‌شود و همین‌طور که می‌بینید ذخیره‌ی عدد ۳۴۵ در byte موجب شده تا عدد ۸۹ در آن قرار گیرد که دقیقاً معادل تفریق ۳۴۵ و ۲۵۶ است.

به مثال زیر دقت کنید:

```
using System;
class Exampe
{
    static void Main()
    {
        double x, y;
        byte b;
        int i;
        char ch;
        uint u;
        short s;
        long l;

        x = 10.0;
        y = 3.0;

        // Cast double to int, fractional component lost.
        i = (int)(x / y);
        Console.WriteLine("Integer outcome of x / y: " + i);
        Console.WriteLine();

        // Cast an int into a byte, no data lost.
        i = 255;
        b = (byte)i;
        Console.WriteLine("b after assigning 255: " + b +
        " -- no data lost.");

        // Cast an int into a byte, data lost.
        i = 257;
        b = (byte)i;
        Console.WriteLine("b after assigning 257: " + b +
        " -- data lost.");
        Console.WriteLine();
```

```
        // Cast a uint into a short, no data lost.
        u = 32000;
        s = (short)u;
        Console.WriteLine("s after assigning 32000: " + s +
        " -- no data lost.");

        // Cast a uint into a short, data lost.
        u = 64000;
        s = (short)u;
        Console.WriteLine("s after assigning 64000: " + s +
        " -- data lost.");
        Console.WriteLine();

        // Cast a long into a uint, no data lost.
        l = 64000;
        u = (uint)l;
        Console.WriteLine("u after assigning 64000: " + u +
        " -- no data lost.");

        // Cast a long into a uint, data lost.
        l = -12;
        u = (uint)l;
        Console.WriteLine("u after assigning -12: " + u +
        " -- data lost.");
        Console.WriteLine();

        // Cast a byte into a char.
        b = 88; // ASCII code for X
        ch = (char)b;
        Console.WriteLine("ch after assigning 88: " + ch);
    }
}
```

خروجی:

```
Integer outcome of x / y: 3

b after assigning 255: 255 -- no data lost.
b after assigning 257: 1 -- data lost.

s after assigning 32000: 32000 -- no data lost.
s after assigning 64000: -1536 -- data lost.

u after assigning 64000: 64000 -- no data lost.
u after assigning -12: 4294967284 -- data lost.

ch after assigning 88: X
```

بهتر است هر قسمت این برنامه را شرح دهیم. در قسمت اول (x / y) به int تبدیل شده است. در این‌جا قسمت اعشاری نتیجه‌ی تقسیم از بین می‌رود. در قسمت بعد عدد صحیح ۲۵۵ را به byte اختصاص داده‌ایم و به این علت که byte توانایی نگه‌داری این مقدار را دارد، هیچ اطلاعاتی از بین نمی‌رود اما با این حال به cast کردن نیاز است چراکه به‌صورت implicit نمی‌توان int را به byte کانورت کرد. در قسمت بعدتر ۲۵۷ را به byte اختصاص داده‌ایم که موجب از بین رفتن اطلاعات می‌شود.

در بقیه‌ی موارد نیز به همین شکل اتفاق می‌افتد. اگر متغیری که قرار است مقداری در آن ریخته شود، گنجایش کافی برای نگه‌داری از آن را داشته باشد هیچ اطلاعاتی از بین نخواهد رفت، در غیر این صورت بخشی از اطلاعات از بین می‌رود.