بسم اللّه الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

عملگرهای منطقی

مدرس : مهندس افشین رفوآ

www.tahlildadeh.com

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 - واحد 7
88146323 - 88446780 - 88146330

عملگرهای منطقی

مقدمه

برنامه در واقع مجموعه ای از دستور ها است که از compiler درخواست می کند شرایط را بسنجد و بر اساس آن شرایط عمل کند. برای بررسی چنین شرایطی، کامپیوتر بخش عظیمی از زمان خود را صرف مقایسه ی مقادیر مختلف می کند. فرایند مقایسه در حقیقت همان عملیات Boolean است که بسته به مقادیری که مقایسه بر پایه ی آن انجام شده نتیجه ی صحیح (true) یا غلط (false) می دهد.

مقایسه بین دو مقدار هم نوع صورت می گیرد ؛ برای مثال، می توان دو عدد، دو کاراکتر، اسامی دو شهر را با هم مقایسه کرد. از سوی دیگر، مقایسه ی بین دو مقدار متفاوت (از نوع مختلف) کاملاً بی معنا می باشد. برای مثال، نمی توان یک شماره تلفن را با سن کسی مقایسه کرد یا یک نوع موسیقی را با فاصله ی بین دو نقطه قیاس کرد. مشابه عملیات حسابی دودویی (binary)، عملیات مقایسه بر اساس دو مقدار صورت می گیرد. بر خلاف عملیات حسابی که نتایج حاصله آن متنوع و متفاوت هستند، نتیجه ی فرایند مقایسه از دو حالت خارج نیست به طوری که نتیجه یا منطقی صحیح (logical true) یا منطقی ناصحیح (logical false) است. در صورتی که مقایسه صحیح باشد، مقدار توکار 1 یا مثبت را به دست می دهد، که منظور همان مقداری است که از 0 بزرگتر باشد. چنانچه مقایسه صحیح یا true نباشد، غلط یا false تلقی می گردد و مقدار آن 0 خواهد بود.

زبان C# مجهز به عملگرهای گوناگونی است که به کمک آن ها می توان هر قسم مقاسیه ای بین مقادیر هم نوع انجام داد. مقادیر نام برده می توانند عددی، از نوع رشته و یا اشیاء باشند (عملیاتی که بر روی اشیا صورت می گیرد در فرایندی به نام operator overloading یا اضافه بارگذاری عملگر، اختصاصی تنظیم می شوند.

هنگام نوشتن دستورات شرطی (دستوراتی که در شرطی ها به کار می روند) یک سری ملزومات اولیه هست که باید رعایت کرد.

سادگی و وضوح : یک دستور باید کاملاً واضح و تا حد ممکن ساده ولی در عین حال کامل باشد. زمانی که دستوری طولانی می گردد، ممکن است به بخش های کوتاه مختلفی تقسیم شود که وضوح دستور را مختل کرده و منجر به بروز مسائل و مشکلات متعددی می شود.

واقعی : دستور مورد نظر باید به عنوان یک حقیقت عرضه شود و نه یک نظر یا عقیده، به این معنا که لزومی ندارد شما آن دستور را دوست داشته باشید بلکه باید مثل اکثریت آن را به عنوان صحیح (true) یا غلط (false) بپذیرید. در حقیقت، نیازی نیست که دستور به خودی خود درست باشد بلکه باید آن را به عنوان صحیح پذیرفت. بر این اساس، دستوری مثل اینکه " یک ساعت 45 دقیقه هست " لزومی ندارد با نظر شما همخوانی داشته باشد بلکه باید یا به عنوان صحیح یا غلط پذیرفته شود. دستوری مثل " این متقاضی شغل گزینه ی مناسبی است " یک نظر می باشد، بنابراین در دستورات شرطی جایی ندارد.

درستی ضمنی و وابسته به موقعیت : زمانی که دستوری نوشته یا ایجاد می شود باید به عنوان صحیح یا غلط پذیرفته شود، اگرچه ممکن است بعده ها تغییر کند. برای مثال، تصور کنید در سال مشخصی دستوری به این صورت نوشته شود " امسال ماه فبریه 28 روز خواهد داشت ". اگرچه استفاده از آن مجاز می باشد، باید تا حد ممکن از به کاربردن آن خودداری کرد، مگر در شرایط اضطراری.

معکوس : باید این امکان وجود داشته باشد که یک دستور عکس خود را پیدا کند. به عبارت روشن تر، زمانی که دستوری نوشته و به عنوان صحیح یا غلط پذیرفته می شود، دستور ضد یا مغایری باید وجود داشته باشد که آن را عکس کند (از غلط به صحیح و بالعکس). برای مثال، چنانچه دستوری دارید مانند " این متقاضی کار 18 سال سن دارد " باید این امکان وجود داشته باشد که بگویید " این متقاضی کار 18 ساله نیست " یا " این متقاضی کار جوان تر از 18 سال به نظر می رسد ".

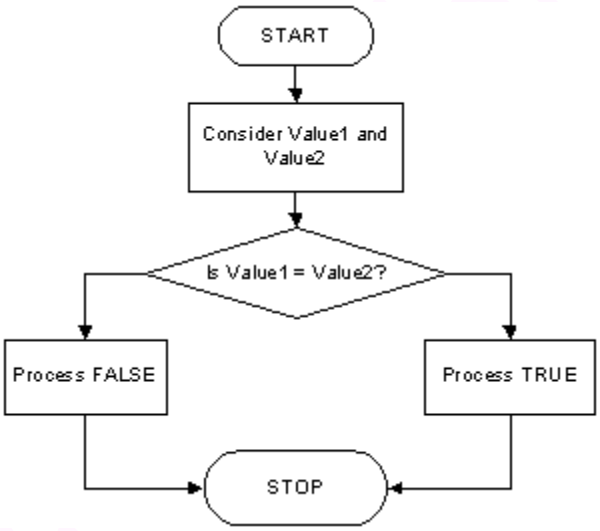
سعی کنید تا حد ممکن دستورات خود را دقیق و واضح بیان و تنظیم کنید. با این کار برنامه های شما خوانا می شوند و عیب زدایی آن آسان می گردد.

عملگر تساوی ==

برای مقایسه ی دو متغیر، C# عملگر == را به کار می برد. فرمول آن به شرح زیر می باشد.

```
value1 == Value2
```

عملیات تساوی به منظور کشف برابری بین دو متغیر (یا یک متغیر و یک ثابت) بکار می رود؛ به عبارت دیگر، ایا دو متغیر مقدار یکسانی دارند یا نه. compiler از طریق دستور نحوی، مقدار value1 و value2 را با هم مقایسه می کند. چنانچه مقدار این دو value با هم یکسان یا برابر بود، عملیات مقایسه نتیجه ی صحیح یا true را به دست می دهد، اما در صورتی که دو متغیر فوق با هم متفاوت باشند (مقدار آن ها باهم برابر نباشد)، مقایسه نتیجه ی غلط یا false را ارائه می دهد.



اغلب مقایسه هایی که در زبان C# صورت می گیرد، برای دستورات شرطی به کار گرفته می شود. نتیجه ی پروسه ی مقایسه را می توان به یک متغیر نیز اختصاص داد. برای ذخیره ی نتیجه ی مقایسه، لازم است عملیات مقایسه را داخل پرانتز قرار دهید.

مثال

```
using System;
public class Exercise
{
            static int Main()
            {
```

```
        var value1 = 15;
        var value2 = 24;
        Console.Write("Value 1 = ");
        Console.WriteLine(value1);
        Console.Write("Value 2 = ");
        Console.WriteLine(value2);
        Console.Write("Comparison of value1 == 15 produces ");
        Console.WriteLine(value1 == 15);
                return 0;
            }
        }
```

نتیجه ی زیر حاصل می گردد.

```
Value 1 = 15
        Value 2 = 24
        Comparison of value1 == 15 produces True
```

لازم است بین عملگر جایگزین " = " و عملگر منطقی تساوی " == " تفاوت قائل شد. عملگر جایگزین برای دادن مقداری جدید به متغیر به کار می رود مانند عدد 244 =. به یاد داشته باشید که عملوندی (operand) که در سمت چپ " = " قرار می گیرد باید همیشه یک متغیر باشد و نه یک ثابت. عملگر " == " هیچگاه برای تخصیص دادن مقدار به کار نمی رود، و در صورت استفاده از آن برای این منظور با error مواجه می شوید. عملگر " == " تنها برای مقایسه ی دو مقدار به کار می رود. عملوندی که در سمت چپ عملگر مزبور قرار می گیرد می تواند متغیر، ثابت و یا یکی متغیر و دیگری ثابت باشد. چنانچه از یک عملگر به طور اشتباه به جای دیگری استفاده کنید هنگام ترجمه ی (compile) برنامه با error مواجه می شوید.

عملگر منطقی Not

پس از تعریف و مقداردهی متغیر (این کار از طریق مقداردهی اولیه یا تغییر مقدار انجام می شود)، در واقع متغیر مذکور در برنامه زنده (یا ایجاد) می شود. حال متغیر می تواند در عملیات لازم شرکت داشته باشد. compiler حساب تمامی متغیرهای موجود در برنامه (و متغیرهایی که در حال پردازش هستند) را دارد. (چنانچه متغیری مورد استفاده قرار نگیرد یا برای پردازش در دسترس نباشد که در برنامه نویسی مجازی غیرفعال تلقی می گردد) برای غیرفعال کردن متغیر (به طور موقت)، باید مقدار آن را nullify

(تخصیص مقدار null به متغیر) کرد. C# متغیری را که مقدار آن تهی (Null) باشد stern در نظر می گیرد. به منظور غیر فعال کردن (غیر قابل استفاده و از دسترس خارج ساختن آن) متغیر، حین توسعه و شکل گیری برنامه، عملگر منطقی Not " ! " را اعمال کنید. ترکیب نحوی آن به صورت زیر می باشد.

```
!Value
```

برای استفاده از عملگر منطقی Not دو گزینه ی اصلی پیش رو دارید. یکی از معمول ترین روش های استفاده از آن، بررسی وضعیت متغیر است.

برای تخصیص مقدار null به متغیر ، کافی است علامت تعجب " ! " را سمت چپ متغیر قرار دهید. مانند مثال زیر

```
using System;
public class Exercise
{
            static int Main()
            {
        bool hasAirCondition = true;
        bool doesIt;
        Console.Write("hasAirCondition = ");
        Console.WriteLine(hasAirCondition);
        doesIt = !hasAirCondition;
        Console.Write("doesIt          = ");
        Console.WriteLine(doesIt);
                return 0;
            }
        }
```

این نتیجه حاصل می گردد.

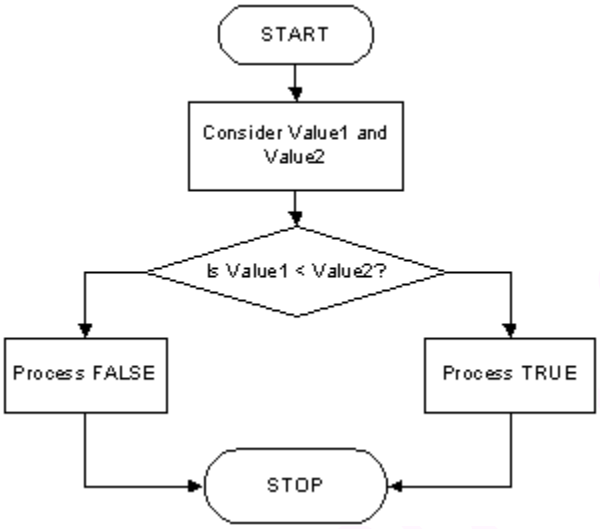
```
hasAirCondition = True
        doesIt          = False
```

زمانی که متغیری مقداری دارد، " زنده " یا فعال تلقی می شود. برای از دسترس خارج ساختن متغیر می توان عملگر " Not " را به آن تخصیص داد. پس از تخصیص مقدار تهی (null) به متغیر، مقدار منطقی آن تغییر می یابد. اگر مقدار منطقی متغیر true یا همان 1 بود، حال به false یا 0 تبدیل می شود. به این ترتیب می توان مقدار منطقی متغیر را معکوس کرد.

کوچکتر از : <

برای کوچکتر نشان دادن مقداری از مقدار دیگر، عملگر < را به کار ببرید. ترکیب نحوی عملگر نام برده به این شکل است.

```
Value1 < Value2
```

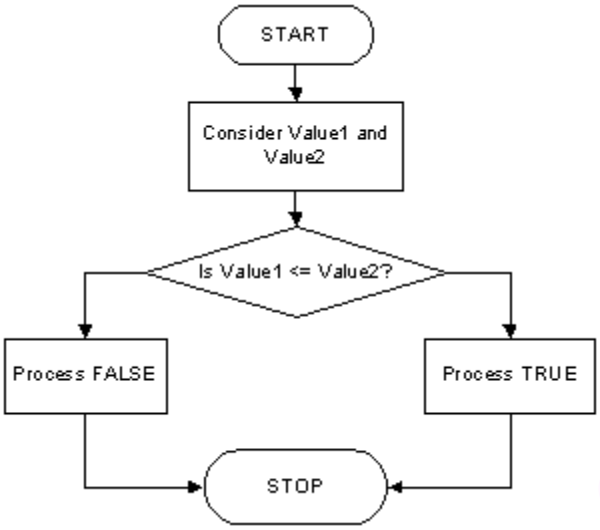


کوچکتر از مساوی : =>

دو عملیات پیشین را می توان با هم ترکیب کرد و با استفاده از آن دو مقدار را با هم مقایسه کرد. با این کار متوجه می شوید که دو مقدار با هم برابرند یا مقدار اولی از مقدار دومی کوچک تر است. عملگر گفته شده به این شکل است : => و دستور نحوی آن به ترتیب زیر می باشد.

```
Value1 <= Value2
```

عملگر => درست مثل دو عمگلر پیشین، عملیات مقایسه انجام می دهد. در صورتی که مقدار value1 و value2 برابر باشد، نتیجه صحیح یا مثبت به دست می آید. چنانچه عملوند طرف چپ، که در این مورد منظور value1 است، مقداری داشته باشد که از مقدار عملوند دیگر، value2، کمتر یا کوچکتر باشد نتیجه باز هم صحیح (true/positive) خواهد بود.

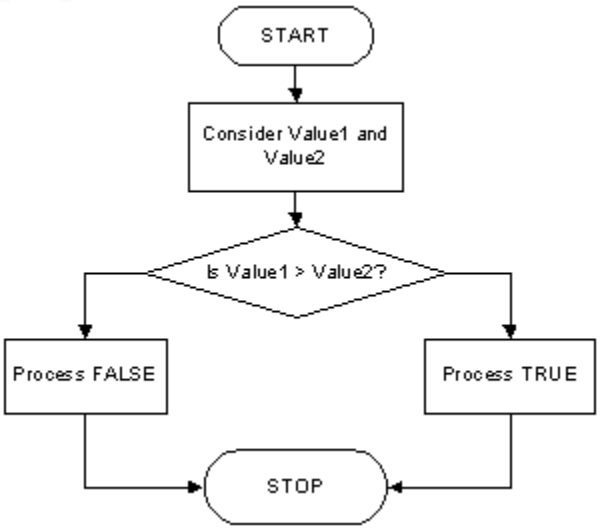


بزرگتر از: >

برای بزرگتر نشان دادن مقداری از مقدار دیگر، عملگر > مورد استفاده قرار می گیرد. فرمول آن به صورت زیر است.

```
Value1 > Value2
```

هر دو عملگر، در این مثال value1 و value2، می توانند متغیر باشند، یا عملگر سمت چپ متغیر و دیگری ثابت باشد. چنانچه مقدار واقع در سمت چپ عملگر بزرگتر از مقدار طرف راست آن بود، مقایسه نتیجه ی صحیح یا مثبت را به دست می دهد. در غیر این صورت، مقایسه نتیجه ی غلط یا تهی (null) ارائه می دهد.



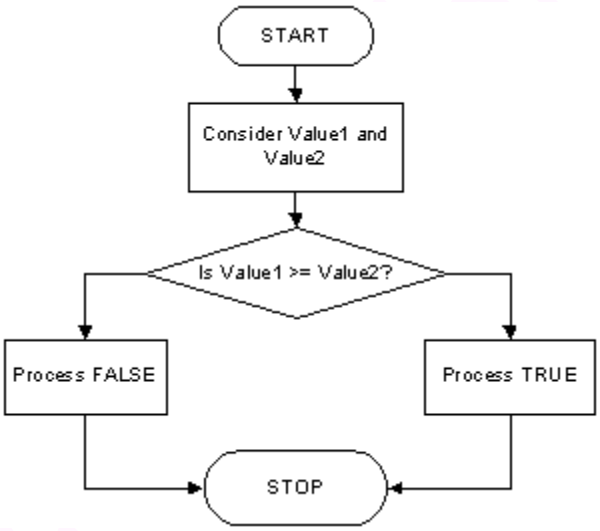
www.tahlildaneh.com

بزرگتر از یا مساوی =>

دو عملگر بزرگتر از و تساوی با هم ترکیب شده و عملگر رو به رو را تولید می کند : =>. به این عملگر " بزرگتر از یا مساوی " می گویند. دستور نحوی آن به صورت زیر است.

```
Value1 >= Value2
```

عملیات مقایسه روی هر دو عملوند value1 و value2 اجرا می شود. چنانچه مقدار هر دو عملوند برابر باشد، مقایسه نتیجه ی صحیح یا مثبت می دهد. در صورتی که مقدار عملوند چپ از مقدار عملوند راست بزرگتر باشد، باز هم نتیجه صحیح (true) خواهد شد. حال اگر مقدار عملوند سمت چپ از مقدار عملوند سمت راست کوچکتر بود، نتیجه ی فرایند غلط یا تهی (null) خواهد شد.



جدول زیر چکیده ی عملگرهای منطقی نام برده را فهرست می کند.

Operator	Meaning	Example	Opposite
==	Equality to	a == b	!=
!=	Not equal to	12 != 7	==
<	Less than	25 < 84	>=
<=	Less than or equal to	Cab <= Tab	>
>	Greater than	248 > 55	<=

>=	Greater than or equal to	Val1 >= Val2	<

www.tahlildadeh.com