فهرست مطالب

عملگرهای اساسی#C

مقدمه

در برنامه نویسی منظور ازoperation ، عملیاتی است که به منظور اصلاح مقدار متغیر موجود، (یا ایجاد مقداری جدید با ترکیب مقدارهای جاری)، روی مقدار معینی انجام می شود. بنابراین، operationبا استفاده از حداقل یک نشانه (symbol) و یک مقدار (value) امکان پذیر می شود. نشانه ای که در عملیات به کار برده شده، (operatorعملگر) خوانده می شود. حال، مقداری که در عملیات نام برده استفاده شده operand (عمل وند) گفته می شود.

Unary operator(عملگر یگانی)، عملگری است که عملیات خود را روی تنها یک عملوند انجام می دهد.چنانچه، عملگری عملیات مورد نیاز را روی دو عملوند انجام دهد، binary operand(عملگر دوگانی) اطلاق می گردد.

معرفی عملگرها و عملوندها

- ابتدا، برنامه ی Microsoft Visual Studio را راه اندازی کنید .
- به منظور ایجاد برنامه ی کاربردی جدید، به فهرست گزینه ی اصلی مراجعه کرده، سپس روی گزینه ی File - >New Project...کلیک کنید .
- گزینه ی Empty Projectرا از فهرست میانی انتخاب کنید .
- اسم را به gdcs4 تغییر داده، روی ok کلیک کنید .
- برای ایجاد فایل ویژه ی کد مورد نظر، main menuرا باز کرده، سپس Project ->Add New Item... را انتخاب کنید .
- Code File را از لیست میانی انتخاب کنید .
- اسم را به CleaningOrder تغییر دهید .
- حال، گزینه ی Add را انتخاب کنید .

این علامت در واقع پرکاربردترین و کارامدترین عملگر در زبان C#می باشد. از این علامت به منظور ایجاد بخشی از یک کد استفاده می شود .به این ترتیب، وظیفه ی آن تعیین حد و مرز و سامان دهی فضای نامی(name space) ، کلاس ها، ساختارها و استثنات است. البته، می توان آن را به دلخواه در دستورهای شرطی به کاربرد. از عملگر مزبور، به منظور ایجاد محدوده ی متغیر (variable scope) نیز استفاده می شود .

نحوه ی استفاده از {}

در فایل خالی که در اختیار شما قرار داده می شود، دستور زیر را وارد کنید .

```
class Order
{
}
```

پرانتز ()

از عملگر پرانتز به منظور جداسازی گروهی از اقلام به کار می رود که متعلق به یک موجودیت (entity) خاص است. برای مثال، با پرانتز می توان متد Mainرا از یک متغیر عادی جدا کرد. مثال زیر را در نظر بگیرید.

```
class Exercise
{
static void Main()
{
}
}
```

از عملگر پرانتز می توان به منظور جداسازی operation (عملیات) یا expression (عبارت) نیز استفاده کرد.

نحوه ی استفاده از عملگر پرانتز

برای به کاربردن این عملگر، اقدامات زیر را انجام دهید.

```
class Order
{
static void Main()
{
}
}
```

عملگر نقطه ویرگول " ; "

علامت نقطه ویرگول، نشانگر اتمام یک عبارت یا تعریف است.

مثال :

```
int number;
```

نحوه ی به کاربردن نقطه ویرگول

در فایل خالی که در اختیار شما قرار داده می شود، دستورات زیر را تایپ کنید.

```
class Order
{
    static void Main()
    {
        doublemondayDiscount;
    }
}
```

عملگر ویرگول " ، "

ویرگول به منظور جداسازی متغیرهای یک گروه به کار می رود .برای مثال، با کمک ویرگول می توان اسم متغیرهایی که با یک نوع داده تعریف شده را از هم جدا کرد. به مثال زیر توجه کنید .

```
class Exercise
{
    static void Main()
    {
        string firstName, lastName, fullName;
    }
}
```

از ویرگول می توان به منظور جداسازی اعضای یک enumeration یا آرگومان های یک متد نیز استفاده کرد.

استفاده از ویرگول

برای این منظور، فایل مربوطه را به صورت زیر تغییر دهید.

```
1
2         class Order
3{
    static void Main()
4   {
5       string customerName, homePhone;
6       uint numberOfShirts, numberOfPants, numberOfDresses;
7       decimal priceOneShirt, priceAPairOfPants, priceOneDress;
8       uint orderMonth, orderDay, orderYear;
9       double mondayDiscount;
    }
10}
11
```

عملگر جایگزین (=)

پس از تعریف متغیر، compilerمقدار مشخصی از حافظه ی کامپیوتر را به آن اختصاص می دهد. حال، آن میزان حافظه تا زمانی که شما آن را با مقدار معینی پر نکرده اید، خالی می ماند. برای قرار دادن مقداری در حافظه ی اختصاص داده شده به متغیر، می توان از عملگر جایگزین " = " بهره گرفت. دستور نحوی آن به این شکل است.

```
1VariableName = Value
```

فاکتور VariableName باید اسم متغیر معتبری باشد. توجه داشته باشید که نمی توان از مقدار عددی (numeric value) یا رشته ای که داخل " "قرار داده شده باشد، برای این منظور استفاده کرد .در مثال زیر یک مقدار عددی به متغیر اختصاص داده.

```
1
2               class Exercise
3{
    static void Main()
4   {
5       decimal salary;
6       // Using the assignment operator
7       salary = 12.55M;
    }
8}
9
```

پس از این که متغیری تعریف شد و مقداری به آن اختصاص داده شد، می توان برای مشاهده ی (نمایش) مقدار آن Write() یا WriteLine() را فراخواند.

مثال :

```
class Exercise
{
    static void Main()
    {
        decimal salary;
        // Using the assignment operator
        salary = 12.55M;
        System.Console.Write("Employee's Hourly Salary: ");
        System.Console.WriteLine(salary);
    }
}
```

نتیجه ی زیر به دست می آید.

```
Employee's Hourly Salary: $12.55
```

کُدِ فوق ابتدا متغیر را تعریف کرده، سپس مقدار را به آن تخصیص داده است.این رویه زمانی دنبال می شود که، بخواهید مقدار یک متغیر را تغییر دهید. از فرایند تخصیص مقداری به متغیر، پس از اینکه متغیر تعریف شده، با نام مقداردهی اولیه یا initializingیاد می شود.

مثال :

```
class Exercise
{
    static void Main()
    {
        // Using the assignment operator
        decimal salary = 12.55M;
        System.Console.Write("Employee's Hourly Salary: ");
        System.Console.WriteLine(salary);
    }
}
```

همان طور که پیش تر ذکر شد، می توان چند متغیر را با استفاده از تنها یک نوع داده همزمان تعریف کرد، سپس اسامی آن ها را با ویرگول از هم جدا کرد. حین انجام این عملیات، همچنین می توان هر متغیر را با value دلخواه و پیش از عملگر ویرگول یا نقطه ویرگول، مقداردهی اولیه کرد. به مثال زیر توجه کنید.

```
class Exercise
{
    static void Main()
    {
```

```
        // Initializing various variables when declaring them with the same data type
        double value1 = 224.58, value2 = 1548.26;
        System.Console.Write("Value 1 = ");
        System.Console.WriteLine(value1);
        System.Console.Write("Value 2 = ");
        System.Console.WriteLine(value2);
        System.Console.WriteLine();
    }
}
```

نتیجه:

```
        Value 1 = 224.58
Value 2 = 1548.26
```

نحوه ی تخصیص مقدار به متغیر

به منظور استفاده از عملگر جایگزین =، فایل مرتبط را این گونه اصلاح کنید.

```
        class Order
{
    static void Main()
    {
        string customerName, homePhone;
        uint numberOfShirts = 5,
        numberOfPants = 2,
        numberOfDresses = 0;
        decimal priceOneShirt = 0.95M,
        priceAPairOfPants = 2.95M,
        priceOneDress = 4.55M;
        uint orderMonth = 3, orderDay = 15, orderYear = 2002;
        double mondayDiscount = 0.25;// 25%;
    }
}
```

علامت (')

از (') به منظور دخیل کردن یک کاراکتر برای مقداردهی اولیه ی متغیری به کار می رود که با نوع داده ی char تعریف شده باشد.

مثال :

```
class Exercise
{
    static void Main()
    {
        char gender;
        string firstName, lastName, fullName;
        gender = 'M';
    }
}
```

نمی توان بیش از یک کاراکتر داخل '' قرار داد. به عبارت دیگر، جمیع نشانه های به کاربرده شده نباید به بیشتر از یک کاراکتر ارزیابی شود.

مثال :

```
class Exercise
{
    static void Main()
    {
        System.Console.WriteLine('\n');
    }
}
```

علامت (")

از این علامت به منظور تعیین حد و مرز یک رشته (string) استفاده می شود. داخل " " می توان یک فضای خالی، یک کاراکتر، یک کلمه یا یک گروه کلمه قرار داد. جمیع این اقلام داخل " "، یک رشته را تشکیل می دهد .به مثال زیر توجه کنید .

```
class Exercise
{
    static void Main()
    {
        System.Console.WriteLine("The Wonderful World of C#!!!");
    }
}
```

استفاده از علامت نقل و قول

- برای این منظور اقدامات زیر را انجام دهید .

```
class Order
{
static void Main()
{
    string customerName = "James Burreck",
    homePhone = "(202) 301-7030";
    uint numberOfShirts = 1,
    numberOfPants = 1,
    numberOfDresses = 1;
    decimal priceOneShirt = 0.95M,
    priceAPairOfPants = 2.95M,
    priceOneDress = 4.55M;
    uint orderMonth = 3, orderDay = 15, orderYear = 2002;
    double mondayDiscount = 0.25;// 25%;
    System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
    System.Console.WriteLine("=========================");
    System.Console.Write("Customer:  ");
    System.Console.WriteLine(customerName);
    System.Console.Write("Home Phone: ");
    System.Console.WriteLine(homePhone);
    System.Console.Write("Order Date: ");
    System.Console.Write(orderMonth);
    System.Console.Write('/');
    System.Console.Write(orderDay);
    System.Console.Write('/');
    System.Console.WriteLine(orderYear);
    System.Console.WriteLine("-------------------------");
    System.Console.WriteLine("Item Type Qty Sub-Total");
    System.Console.WriteLine("-----------------------");
    System.Console.Write("Shirts     ");
    System.Console.Write(numberOfShirts);
    System.Console.Write("    ");
    System.Console.WriteLine(priceOneShirt);
    System.Console.Write("Pants      ");
    System.Console.Write(numberOfPants);
    System.Console.Write("    ");
    System.Console.WriteLine(priceAPairOfPants);
    System.Console.Write("Dresses    ");
    System.Console.Write(numberOfDresses);
    System.Console.Write("    ");
    System.Console.WriteLine(priceOneDress);
    System.Console.WriteLine("-------------------------");
    System.Console.Write("Monday Discount: ");
    System.Console.Write(mondayDiscount);
    System.Console.WriteLine('%');
    System.Console.WriteLine("=========================");
    System.Console.ReadKey();
}
}
```

- برای مشاهده ی نتیجه برنامه را اجرا کنید. این نتیجه حاصل می گردد .

```
1
2            -/- Georgetown Cleaning Services -/-
3========================
4Customer:  James Burreck
5Home Phone: (202) 301-7030
6Order Date: 3/15/2002
7------------------------
8Item Type Qty Sub-Total
9------------------------
10Shirts     1    0.95
 Pants      1    2.95
11Dresses    1    4.55
12------------------------
13Monday Discount: 0.25%
14========================
```

- حال، با زدن کلید Enter به محیط برنامه نویسی بازگردید.

کروشه []

از این علامت بیشتر زمان ها برای مدیریت و کنترل اندیس یا بعد یک آرایه استفاده می شود. درباره ی کاربرد این عملگر در بخش آرایه ها بحث می کنیم .

عملگر مثبت (+)

علم جبر برای دسته بندی ارقام از یک جور خط کش استفاده می کند. وسط این خط کش عدد ۰ قرار دارد. حال، ارقامی که در سمت چپ صفر قرار می گیرند منفی تلقی می شوند و ارقامی که در سمت راست جای گرفته اند مثبت.

-∞		-6	-5	-4	-3	-2	-1		1	2	3	4	5	6		+∞
0																
-∞		-6	-5	-4	-3	-2	-1		1	2	3	4	5	6		+∞

مقداری که در سمت راست ۰ قرار دارد مثبت محسوب می شود .اعداد مثبت با علامت + نمایش داده می شوند4+ : ، 228+، .90335+در این مورد، + در واقع یک عملگر یگانی (unary operator) حساب می شود، زیرا بر تنها یک عملوند (operand)تاثیر می گذارد. توجه داشته باشید که عملگر + باید همیشه سمت چپ عملوند قرار گیرد و نه سمت راست آن.

البته، در علم ریاضی این علامت نشان داده نمی شود، و اگر عددی بدون هیچگونه علامت خاص نوشته شود، به طور معمول مثبت تلقی می گردد .پس ارقام +۴، +۲۲۸و 90335+را می توان به این صورت نیز نوشت : ۴، .228،90335به خاطر این که مقدار فوق هیچ گونه علامتی را به یدک نمی کشد، آن راunsigned ، یا بدون علامت می خوانند.

مثال :

```
1
2         class Exercise
3{
4     static void Main()
5     {
6         // Displaying an unsigned number
7         System.Console.Write("Number = ");
8         System.Console.WriteLine(+802);
9     }
10}
11
```

نتیجه

```
1Number = 802
```

عملگر منفی (-)

همان طور که در خط کش بالا مشاهده می کنید، برای نشان دادن هر عددی که در سمت چپ صفر قرار می گیرد، مجبور به استفاده از علامت - هستیم12- : ، 448-، .32706-عددی که با این علامت همراه باشد منفی اطلاق می شود. برخلاف عدد مثبت، عدد منفی باید با نشان مناسب (-) همراه باشد. برای منفی کردن یک عدد مثبت هم کافی است فقط یک علامت منفی به آن اضافه کنید.

```
1
2         class Exercise
3{
4     static void Main()
5     {
6         // Displaying an unsigned number
7         System.Console.Write("First Number ");
8         System.Console.WriteLine(+802);
9         // Displaying a negative number
10        System.Console.Write("Second Number ");
11        System.Console.WriteLine(-802);
12    }
13}
14
```

نتیجه

```
1        First Number 802
2Second Number -802
```

عملگرهای یگانی : اندازه ی عملگر

زبان C# عملگر یگانی sizeof را برای محاسبه ی مقدار حافظه ی مورد نیاز یک نوع داده، در اختیار برنامه نویس قرار می دهد .

به منظور محاسبه مقدار حافظه ی مورد نیاز یک نوع داده ی خاص، باید آن (data type) را داخل پرانتز عملگر مذکور قرار داد.

مثال :

```
1
2        class Exercise
3{
4    static void Main()
5    {
6        double period = 155.50;
7        int size = sizeof(double);
8        System.Console.Write("The value ");
9        System.Console.Write(period);
10       System.Console.Write(" uses ");
11       System.Console.Write(size);
12       System.Console.WriteLine(" bytes\n");
13   }
14}
15
```

نتیجه :

```
1The value 155.5 uses 8 bytes
```

زیر نمونه های بیشتری را مشاهده می کنید .

```
1            class Exercise
2{
3    static void Main()
4    {
5        // Thesizeof operator used to get the memory size used by
6        // a variable declared with a certain a data type
7        System.Console.WriteLine("The sizeof Operator");
8        System.Console.WriteLine("==========================");
```

```
        System.Console.WriteLine("Data Type    Memory Size");
        System.Console.WriteLine("-------------------------");
        System.Console.Write("char          ");
        System.Console.Write(sizeof(char));
        System.Console.WriteLine(" Bytes");
        System.Console.Write("bool          ");
        System.Console.Write(sizeof(bool));
        System.Console.WriteLine(" Bytes");
        System.Console.Write("int           ");
        System.Console.Write(sizeof(int));
        System.Console.WriteLine(" Bytes");
        System.Console.Write("uint          ");
        System.Console.Write(sizeof(uint));
        System.Console.WriteLine(" Bytes");
        System.Console.Write("short         ");
        System.Console.Write(sizeof(short));
        System.Console.WriteLine(" Bytes");
        System.Console.Write("ushort        ");
        System.Console.Write(sizeof(ushort));
        System.Console.WriteLine(" Bytes");
        System.Console.Write("byte          ");
        System.Console.Write(sizeof(byte));
        System.Console.WriteLine(" Bytes");
        System.Console.Write("sbyte         ");
        System.Console.Write(sizeof(sbyte));
        System.Console.WriteLine(" Bytes");
        System.Console.Write("float         ");
        System.Console.Write(sizeof(float));
        System.Console.WriteLine(" Bytes");
        System.Console.Write("double        ");
        System.Console.Write(sizeof(double));
        System.Console.WriteLine(" Bytes");
        System.Console.Write("decimal       ");
        System.Console.Write(sizeof(decimal));
        System.Console.WriteLine(" Bytes");
        System.Console.Write("long          ");
        System.Console.Write(sizeof(long));
        System.Console.WriteLine(" Bytes");
        System.Console.Write("ulong         ");
        System.Console.Write(sizeof(ulong));
        System.Console.WriteLine(" Bytes");
        System.Console.WriteLine("=========================");
        System.Console.WriteLine();
    }
}
```

نتیجه

```
                The sizeof Operator
=========================
Data Type    Memory Size
-------------------------
char      2 Bytes
bool      1 Bytes
int       4 Bytes
uint      4 Bytes
short     2 Bytes
```

```
ushort     2 Bytes
byte       1 Bytes
sbyte      1 Bytes
float      4 Bytes
double     8 Bytes
decimal    16 Bytes
long       8 Bytes
ulong      8 Bytes
==========================

```

عمل جمع

زمانی که بخواهیم دو چیز هم نوع را به هم اضافه کنیم (هر تعداد دفعه ای که لازم باشد)، از addition operation یا عمل جمع بهره می گیریم. گاهی نیز، به جای اضافه کردن یک چیز به چیز دیگر، یک گروه را به گروه دیگر اضافه می کنیم. تنها تفاوت آن در سرعت انجام عملیات است. در ریاضی برای انجام فرایند جمع از علامت + استفاده می شود. در C# نیز همین علامت کاربرد دارد .

برای به دست آوردن حاصل جمع دو مقدار، باید اولی را به دومی اضافه کرد. مقدار جدیدی در اختیار شما قرار می گیرد، به این صورت

```
value3 = value1 + value2
```

در مورد ارقام، نحوه و ترتیب قرارگیری اعداد چندان اهمیتی ندارد. برای مثال، value1 + value2هیچ فرقی با value2 + value1ندارد.

مثال :

```

class Exercise
{
    static void Main()
    {
        System.Console.Write("244 + 835 = ");
        System.Console.WriteLine(244 + 835);
    }
}

```

نتیجه

```
244 + 835 = 1079
```

همچنین می توان مقدارهایی که از پیش تعریف یا مقداردهی اولیه شده را در برنامه ی خود به کار برد، حتی می توان مقدارها را از کاربر دریافت کرد.

در#C ، می توان برای ایجاد یک رشته ی جدید، متغیرهای رشته را به آن اضافه کرد. عملیات همان طوری که جمع اعداد صورت می گیرد، انجام می شود. برای مثال، "Pie" + "Chart"نتیجه می دهد ."PieChart"به همین ترتیب، می توانید هر تعداد رشته که مایلید با دخیل کردن عملگر + به دستور اضافه کنید.


مثال :

```
1
2        class Exercise
{
3    static void Main()
4    {
5        varfirstName = "Alexander";
6        varlastName = "Kallack";
7        varfullName = firstName + " " + lastName;
8        System.Console.Write("Full Name: ");
9        System.Console.WriteLine(fullName);
    }
}
```

نتیجه ی زیر حاصل می گردد.

```
1Full Name: Alexander Kallack
2Press any key to continue...
```


استفاده از عملگر +

- فایل را به صورت زیر تغییر دهید.

```
1        class Order
2{
3    static void Main()
4    {
5        string customerName = "James Burreck",
6        homePhone = "(202) 301-7030";
7        uint numberOfShirts = 1,
8        numberOfPants = 1,
9        numberOfDresses = 1;
10       uint totalNumberOfItems;
11       decimal priceOneShirt = 0.95M,
12       priceAPairOfPants = 2.95M,
         priceOneDress = 4.55M;
         uint orderMonth = 3, orderDay = 15, orderYear = 2002;
```

```
        totalNumberOfItems = numberOfShirts + numberOfPants +
numberOfDresses;
        System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
        System.Console.WriteLine("========================");
        System.Console.Write("Customer:  ");
        System.Console.WriteLine(customerName);
        System.Console.Write("Home Phone: ");
        System.Console.WriteLine(homePhone);
        System.Console.Write("Order Date: ");
       System.Console.Write(orderMonth);
        System.Console.Write('/');
        System.Console.Write(orderDay);
        System.Console.Write('/');
        System.Console.WriteLine(orderYear);
        System.Console.WriteLine("------------------------");
        System.Console.WriteLine("Item Type Qty Sub-Total");
        System.Console.WriteLine("------------------------");
        System.Console.Write("Shirts     ");
        System.Console.Write(numberOfShirts);
        System.Console.Write("    ");
        System.Console.WriteLine(priceOneShirt);
        System.Console.Write("Pants      ");
        System.Console.Write(numberOfPants);
        System.Console.Write("    ");
        System.Console.WriteLine(priceAPairOfPants);
        System.Console.Write("Dresses    ");
        System.Console.Write(numberOfDresses);
        System.Console.Write("    ");
        System.Console.WriteLine(priceOneDress);
        System.Console.WriteLine("------------------------");
        System.Console.Write("Number of Items: ");
        System.Console.WriteLine(totalNumberOfItems);
        System.Console.WriteLine("========================");
        System.Console.ReadKey();
    }
}
```

- حال، برنامه را اجرا کنید. نتیجه ی زیر به دست می آید .

```

-/- Georgetown Cleaning Services -/-
========================
Customer:  James Burreck
Home Phone: (202) 301-7030
Order Date: 3/15/2002
------------------------
Item Type Qty Sub-Total
------------------------
Shirts     1    0.95
Pants      1    2.95
Dresses    1    4.55
------------------------
Number of Items: 3
========================

```

- پنجره ی DOS را بسته، به محیط برنامه نویسی خود بازگردید .

افزایش دادن متغییر

آسان ترین روش به منظور افزایش یک مقدار، اضافه کردن ۱ به آن است. پس از افزودن ۱ به آن، متغیر یا مقدار برای همیشه تغییر پیدا کرده، و متغیر حاوی مقدار جدید خواهد بود. مثال زیر این نمونه را به زیبایی نمایش داده است.

```
1
2// This program studies value incrementing
3public class Exercise
4{
5    static void Main()
6    {
7        var value = 12;
8        System.Console.WriteLine("Techniques of incrementing a value");
9        System.Console.Write("Value = ");
10       System.Console.WriteLine(value);
11       value = value + 1;
12       System.Console.Write("Value = ");
13       System.Console.WriteLine(value);
14   }
15}
```

نتیجه فرایند

```
1Techniques of incrementing a value
2Value = 12
3Value = 13
```

زبان C# برای این منظور، عملگر افزایش "++" (increment operator) را برای برنامه نویس فراهم می کند. به جای نوشتن Value = Value + 1 ، می توان نوشت Value++و نتیجه نیز همان خواهد بود. برنامه فوق را به این ترتیب نیز می توان نوشت.

```
1       // This program studies value incrementing
2public class Exercise
3{
4    static void Main()
5    {
6        var value = 12;
7
8        System.Console.WriteLine("Techniques of incrementing a value");
9        System.Console.Write("Value = ");
10       System.Console.WriteLine(value);
11       value++;
12       System.Console.Write("Value = ");
13       System.Console.WriteLine(value);
14   }
15}
```

```
14
15
```

عملگر ++ نیز یک عملگر یگانی محسوب می شود، زیرا تنها یک متغیر را دستخوش تغییر خود قرار می دهد.این عملگر با اضافه کردن ۱ به متغیر، آن را تغییر می دهد. هر بار که Value++ اجرا می شود، compilerمقدار قبلی متغیر را گرفته و به آن ۱ اضافه می کند. متغیر نیز سر انجام ، مقدار افزایش یافته را در خود ذخیره می کند.

```
1
2
3         // This program studies value incrementing
4public class Exercise
5{
6     static void Main()
7     {
8          var value = 12;
9          System.Console.WriteLine("Techniques of incrementing a value");
10         value++;
11         System.Console.Write("Value = ");
12         System.Console.WriteLine(value);
13         value++;
14         System.Console.Write("Value = ");
15         System.Console.WriteLine(value);
16         value++;
17         System.Console.Write("Value = ");
18         System.Console.WriteLine(value);
     }
}
```

نتیجه ی زیر به دست می آید.

```
1Techniques of incrementing a value
2Value = 13
3Value = 14
4Value = 15
```

Pre and post increment

موقعیتی که عملگر در آن قرار می گیرد، به خصوص در رابطه با متغیری که اصلاح می کند، بسیار حائز اهمیت است. به منظور افزایش مقدار یک متغیر، پیش از استفاده ی دوباره از آن، باید عملگر را به طور حتم سمت چپ متغیر قرار داد.

```
1      // This program studies value incrementing
2public class Exercise
3{
4     static void Main()
5     {
6          var value = 12;
7          System.Console.WriteLine("Techniques of incrementing a value");
```

```
        System.Console.Write("Value = ");
        System.Console.WriteLine(value);
        System.Console.Write("Value = ");
        System.Console.WriteLine(++value);
        System.Console.Write("Value = ");
        System.Console.WriteLine(value);
    }
}
```

نتیجه ی زیر حاصل می گردد.

```
Techniques of incrementing a value
Value = 12
Value = 13
Value = 13
```

جمع مرکب (compound addition)

افزودن یک مقدار ثابت به متغیر امری غیر عادی نیست. کافی است متغیری که قرار است میزبان مقدار جدید باشد را تعریف کنید.

مثال :

```
// This program studies value incrementing and decrementing

public class Exercise
{
    static void Main()
    {
        double value = 12.75;
        double newvalue;
        System.Console.WriteLine("Techniques of incrementing and
decrementing a value");
        System.Console.Write("Value = ");
        System.Console.WriteLine(value);
        newvalue = value + 2.42;
        System.Console.Write("Value = ");
        System.Console.WriteLine(newvalue);
    }
}
```

نتیجه ی زیر را به دست می دهد.

```
Techniques of incrementing and decrementing a value
Value = 12.75
Value = 15.17
```

روشی که بالا به کار رفته، ایجاب می کند شما در برنامه ی خود دو متغیر داشته باشید. مزیت آن این است که هر متغیر می تواند مقدار خود را داشته باشد، با این وجود مقدار متغیردوم وابسته است به تغییراتی که به متغییر اصلی وارد می شود. گاهی اوقات نیازی به حفظ کردن مقدار اولیه متغیر اصلی نیست، و لازم است که مقدار یک متغیر را به صورت دائمی تغییر دهید . برای این منظور، باید عملیات جمع را مستقیماً روی خود متغیر پیاده کرد (با اضافه کردن مقدار دلخواه به متغیر). با این کار نه تنها مقدار متغیر اصلاح می شود، بلکه دیگر نیازی به متغیر اضافه بر سازمان نیست.

به منظور افزودن مقدار به متغیر و اصلاح آن، می توان عملگر جایگزین را با عملگر + ترکیب کرد و عملگری جدید به وجود آورد.+= :

مثال :

```
1
2       // This program studies value incrementing and decrementing
3public class Exercise
4{
5    static void Main()
6    {
7        var value = 12.75;
8        System.Console.WriteLine("Techniques of incrementing and
decrementing a value");
9        System.Console.Write("Value = ");
10       System.Console.WriteLine(value);
11       value += 2.42;
12       System.Console.Write("Value = ");
13       System.Console.WriteLine(value);
14   }
15}
16
```

برنامه همان نتیجه ی قبلی را به دست می دهد.

عملیات ضرب

مقدمه

ضرب به شما اجازه می دهد یک عدد را تعداد دفعات مشخص (که توسط مقدار دومی مشخص می شود) به خود آن عدد اضافه کنید. برای مثال، به جای این که مقداری را به این صورت به آن اضافه کنیدA + A + A + A : ، می توان ابتدا تعداد دفعاتی که این مقدار در عملیات جمع به کار رفته را پیدا کرده، سپس آن مقدار را در تعداد دفعات تکرار، ضرب کنید که در این مثال : همان عدد 4می باشد .

درست مثل عمل جمع، ضرب نیز شرکت پذیر است .a * b * c = c * b * a : دستور نحوی عمل ضرب از همان قوانین عمل جمع پیروی می کند .

```
1
2class Exercise
3{
4    static void Main()
5    {
6        // Initializing various variables when declaring them with the
7same data type
8        double value1 = 224.58, value2 = 1548.26;
9        var result = value1 * value2;
10       System.Console.Write(value1);
11       System.Console.Write(" * ");
12       System.Console.Write(value2);
13       System.Console.Write(" = ");
14       System.Console.WriteLine(result);
15       System.Console.WriteLine();
16   }
17}
```

نتیجه

```
1224.58 * 1548.26 = 347708.2308
2Press any key to continue...
```

به کاربردن عملگر ضرب

- برای این منظور، فایل مرتبط را به این صورت اصلاح کنید .

```
1    class Order
2{
3    static void Main()
4    {
5        const decimal priceOneShirt = 0.95M;
6        const decimal priceAPairOfPants = 2.95M;
7        const decimal priceOneDress = 4.55M;
8        string customerName = "James Burreck",
9        homePhone = "(202) 301-7030";
10       uint numberOfShirts = 5,
11       numberOfPants = 2,
12       numberOfDresses = 3;
13       uint totalNumberOfItems;
14       decimal subTotalShirts, subTotalPants, subTotalDresses;
15       decimal totalOrder;
16       uint orderMonth = 3, orderDay = 15, orderYear = 2002;
17       totalNumberOfItems = numberOfShirts + numberOfPants +
18numberOfDresses;
```

```
        subTotalShirts = priceOneShirt * numberOfShirts;
        subTotalPants = priceAPairOfPants * numberOfPants;
        subTotalDresses = numberOfDresses * priceOneDress;
        totalOrder = subTotalShirts + subTotalPants + subTotalDresses;
        System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
        System.Console.WriteLine("====================================");
        System.Console.Write("Customer:  ");
        System.Console.WriteLine(customerName);
        System.Console.Write("Home Phone: ");
        System.Console.WriteLine(homePhone);
        System.Console.Write("Order Date: ");
        System.Console.Write(orderMonth);
        System.Console.Write('/');
        System.Console.Write(orderDay);
        System.Console.Write('/');
        System.Console.WriteLine(orderYear);
        System.Console.WriteLine("------------------------------------");
        System.Console.WriteLine("Item Type Qty Unit/Price Sub-Total");
        System.Console.WriteLine("------------------------------------");
        System.Console.Write("Shirts     ");
       System.Console.Write(numberOfShirts);
        System.Console.Write("    ");
        System.Console.Write(priceOneShirt);
        System.Console.Write("    ");
        System.Console.WriteLine(subTotalShirts);
        System.Console.Write("Pants      ");
        System.Console.Write(numberOfPants);
        System.Console.Write("    ");
        System.Console.Write(priceAPairOfPants);
        System.Console.Write("    ");
        System.Console.WriteLine(subTotalPants);
        System.Console.Write("Dresses    ");
        System.Console.Write(numberOfDresses);
        System.Console.Write("    ");
        System.Console.Write(priceOneDress);
        System.Console.Write("    ");
        System.Console.WriteLine(subTotalDresses);
        System.Console.WriteLine("------------------------------------");
        System.Console.Write("Number of Items: ");
        System.Console.WriteLine(totalNumberOfItems);
        System.Console.Write("Total Order:    ");
        System.Console.WriteLine(totalOrder);
        System.Console.WriteLine("====================================");
        System.Console.ReadKey();
    }
}
```

- برنامه را طبق دستور العمل اجرا کرده تا نتیجه ی آن را مشاهده کنید .

```
-/- Georgetown Cleaning Services -/-
===================================
Customer:  James Burreck
Home Phone: (202) 301-7030
Order Date: 3/15/2002
-----------------------------------
Item Type Qty Unit/Price Sub-Total
-----------------------------------
Shirts     5    0.95    4.75
Pants      2    2.95    5.90
Dresses    3    4.55    13.65
-----------------------------------
Number of Items: 10
Total Order:    24.30
===================================
```

- اکنون، پنجره ی DOS را بسته و به محیط برنامه نویسی خود بازگردید .

ضرب مرکب

دیدیم که می توان مقداری را ازمتغیر کم کرد یا مقداری به آن اضافه کرد، سپس نتیجه آن را به متغیر دیگری اختصاص داد. همین عملیات را می توان برای ضرب هم پیاده کرد .

مثال :

```
                        class Exercise
                                {
                      static void Main()
                                {
                    double value = 12.75;
        System.Console.Write("Value = ");
         System.Console.WriteLine(value);
                   value = value * 2.42;
        System.Console.Write("Value = ");
         System.Console.WriteLine(value);
                                }
                                }
```

نتیجه

```
        Value = 12.75
Value = 30.855
Press any key to continue...
```

3

به منظور آسان سازی عملیات، زبان C# عملگر ضرب مرکب (که با این علامت *= نشان داده می شود) را در اختیار برنامه نویس قرار می دهد. به مثال زیر توجه کنید .

```
1
2       class Exercise
3{
4       static void Main()
5       {
6           double value = 12.75;
7           System.Console.Write("Value = ");
8           System.Console.WriteLine(value);
9           value *= 2.42;
10          System.Console.Write("Value = ");
11          System.Console.WriteLine(value);
12      }
13}
14
```

عملگرهای اساسی C#

عملیات تفریق

مقدمه

از این عمل به منظور کم کردن مقداری از مقدار دیگر استفاده می شود. کم کردن متضاد عمل جمع محسوب می شود. عمل تفریق همان طور که می دانید با عملگر (-) انجام می شود. توجه خود را به مثال زیر جلب کنید .

```
1
2       class Exercise
3{
4       static void Main()
5       {
6           // Values used in this program
7           double value1 = 224.58, value2 = 1548.26;
8           var result = value1 - value2;
9           System.Console.Write(value1);
10          System.Console.Write(" - ");
11          System.Console.Write(value2);
12          System.Console.Write(" = ");
13          System.Console.WriteLine(result);
14      }
15}
16
```

نتیجه

```
224.58 - 1548.26 = -1323.68
Press any key to continue...
```

برخلاف عمل جمع، تفریق شرکت پذیر نیست. به عبارت دیگر، a - b –c با a – b - c یکی نیست. برنامه ی زیر این حقیقت را به خوبی نشان می دهد .

```
class Exercise
{
    static void Main()
    {
        // This tests whether the addition is associative
        System.Console.WriteLine(" =+= Addition =+=");
        System.Console.Write("128 + 42 +  5 = ");
        System.Console.WriteLine(128 + 42 + 5);
        System.Console.Write(" 5 + 42 + 128 = ");
        System.Console.WriteLine(5 + 42 + 128);
        System.Console.WriteLine();
        // This tests whether the subtraction is associative
        System.Console.WriteLine(" =-= Subtraction =-=");
        System.Console.Write("128 - 42 -  5 = ");
        System.Console.WriteLine(128 - 42 - 5);
        System.Console.Write(" 5 - 42 - 128 = ");
        System.Console.WriteLine(5 - 42 - 128);
        System.Console.WriteLine();
    }
}
```

نتیجه ی زیر به دست می آید .

```
      =+= Addition =+=
128 + 42 +  5 = 175
 5 + 42 + 128 = 175
 =-= Subtraction =-=
128 - 42 -  5 = 81
 5 - 42 - 128 = -165
```

همان طور که مشاهده می کنید، هر دو عملیات یک نتیجه ی واحد و یکسان را ارائه می دهند. در بخش منها عملیات، اعداد همان ترتیب را دنبال می کنند، ولی نتایج متفاوتی به دست می دهند .

به کاربردن عملگر منها

- برای انجام عمل منها، فایل را به صورت زیر تغیر دهید .

```
class Order
{
    static void Main()
    {
        const decimal priceOneShirt = 0.95M;
        const decimal priceAPairOfPants = 2.95M;
        const decimal priceOneDress = 4.55M;
        const decimal salestaxRate = 0.0575M;// 5.75%
        string customerName = "James Burreck",
        homePhone = "(202) 301-7030";
        uint numberOfShirts = 5,
        numberOfPants = 2,
        numberOfDresses = 3;
        uint totalNumberOfItems;
        decimal subTotalShirts, subTotalPants, subTotalDresses;
        decimal taxAmount, totalOrder, netPrice;
        uint orderMonth = 3, orderDay = 15, orderYear = 2002;
        totalNumberOfItems = numberOfShirts + numberOfPants +
numberOfDresses;
        subTotalShirts = priceOneShirt * numberOfShirts;
        subTotalPants = priceAPairOfPants * numberOfPants;
        subTotalDresses = numberOfDresses * priceOneDress;
        totalOrder = subTotalShirts + subTotalPants + subTotalDresses;
        taxAmount = totalOrder * salestaxRate;
        netPrice = totalOrder - taxAmount;
        System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
        System.Console.WriteLine("=================================");
        System.Console.Write("Customer:  ");
        System.Console.WriteLine(customerName);
        System.Console.Write("Home Phone: ");
        System.Console.WriteLine(homePhone);
        System.Console.Write("Order Date: ");
        System.Console.Write(orderMonth);
        System.Console.Write('/');
        System.Console.Write(orderDay);
        System.Console.Write('/');
        System.Console.WriteLine(orderYear);
        System.Console.WriteLine("---------------------------------");
        System.Console.WriteLine("Item Type Qty Unit/Price Sub-Total");
        System.Console.WriteLine("---------------------------------");
        System.Console.Write("Shirts     ");
        System.Console.Write(numberOfShirts);
        System.Console.Write("    ");
        System.Console.Write(priceOneShirt);
        System.Console.Write("    ");
        System.Console.WriteLine(subTotalShirts);
        System.Console.Write("Pants      ");
        System.Console.Write(numberOfPants);
        System.Console.Write("    ");
        System.Console.Write(priceAPairOfPants);
        System.Console.Write("    ");
        System.Console.WriteLine(subTotalPants);
        System.Console.Write("Dresses    ");
        System.Console.Write(numberOfDresses);
        System.Console.Write("    ");
        System.Console.Write(priceOneDress);
        System.Console.Write("    ");
        System.Console.WriteLine(subTotalDresses);
        System.Console.WriteLine("---------------------------------");
        System.Console.Write("Number of Items: ");
```

```
50          System.Console.WriteLine(totalNumberOfItems);
51           System.Console.Write("Total Order:    ");
52           System.Console.WriteLine(totalOrder);
53           System.Console.Write("Tax Rate:       ");
54           System.Console.Write(salestaxRate * 100);
55           System.Console.WriteLine('%');
56           System.Console.Write("Tax Amount:     ");
57           System.Console.WriteLine(taxAmount);
58           System.Console.Write("Net Price:      ");
59           System.Console.WriteLine(netPrice);
60           System.Console.WriteLine("==================================");
61           System.Console.ReadKey();
62       }
63   }
64
```

- برنامه را اجرا کنید. نتیجه ی زیر به دست می آید .

```
1
2
3            -/- Georgetown Cleaning Services -/-
4==================================
5Customer:  James Burreck
6Home Phone: (202) 301-7030
7Order Date: 3/15/2002
8----------------------------------
9Item Type Qty Unit/Price Sub-Total
10----------------------------------
11Shirts     5    0.95     4.75
12Pants      2    2.95     5.90
13Dresses    3    4.55     13.65
14----------------------------------
15Number of Items: 10
16Total Order:     24.30
17Tax Rate:        5.7500%
18Tax Amount:      1.397250
19Net Price:       22.902750
20==================================
21
22
```

- پنجره ی DOS را بسته و به محیط برنامه نویسی بازگردید .

کاهش دادن متغیر

زمانی که اعداد را به صورت معکوس می شمارید، مثل 8، 7، 6، 5، در واقع یک ۱ رقم از آن کسر می کنید. از این عملیات با نام کاهش مقدار (decrementing a value)یاد می شود .

```
1       // This program studies value decrementing
2public class Exercise
3{
```

```
    static void Main()
    {
        var value = 12;
        System.Console.WriteLine("Techniques of decrementing a value");
        System.Console.Write("Value = ");
        System.Console.WriteLine(value);
        value = value - 1;
        System.Console.Write("Value = ");
        System.Console.WriteLine(value);
    }
}
```

نتیجه :

```
   Techniques of decrementing a value
                          Value = 12
                          Value = 11
```

زبان برنامه نویسی#C ، برای این منظور عملگر کاهش (--) را ارائه می دهد. با استفاده از عملگر مذکور، عملیات فوق را می توان به این صورت اصلاح کرد .

```
      public class Exercise
{
    static void Main()
    {
        var value = 12;
        System.Console.WriteLine("Techniques of decrementing a value");
        System.Console.Write("Value = ");
        System.Console.WriteLine(value);
        value--;
        System.Console.Write("Value = ");
        System.Console.WriteLine(value);
    }
}
```

Pre-decrementing a variable

در این مورد نیز موقعیت عملگر (جایی که در آن قرار می گیرد) اهمیت به سزایی دارد. چنانچه مایلید متغیر را پیش از فراخوانی آن کاهش دهید، باید عملگر کاهش (decrement operator) را سمت چپ عملوند قرار دهید. برنامه ی زیر این عملیات را به تصویر کشیده .

```
1
2       // This program studies value decrementing
3  public class Exercise
4  {
5      static void Main()
6      {
7          var value = 12;
8          System.Console.WriteLine("Techniques of decrementing a value");
9          System.Console.Write("Value = ");
10         System.Console.WriteLine(value);
11         System.Console.Write("Value = ");
12         System.Console.WriteLine(--value);
13         System.Console.Write("Value = ");
14         System.Console.WriteLine(value);
15     }
16 }
```

نتیجه :

```
1 Techniques of decrementing a value
2 Value = 12
3 Value = 12
4 Value = 11
```

تفریق مرکب (compound subtraction)

به منظور کسر مقدار ثابتی (constant value) از یک متغیر، از عملگر -= استفاده می شود .

مثال :

```
1
2       // This program studies value incrementing and decrementing
3   public class Exercise
4   {
5       static void Main()
6       {
7           var value = 12.75;
8           System.Console.WriteLine("Techniques of incrementing and
  decrementing a value");
9           System.Console.Write("Value = ");
10          System.Console.WriteLine(value);
11          value -= 2.42;
12          System.Console.Write("Value = ");
13          System.Console.WriteLine(value);
14      }
15  }
16
```

نتیجه :

```
Techniques of incrementing and decrementing a value
Value = 12.75
Value = 10.33
```

عملیات تقسیم (division operation)

مقدمه

تقسیم یک مقدار عبارتند از جداسازی آن به بخش های متعدد .برای مثال، هنگامی که سیبی را از وسط نصف می کنید، در واقع آن را به دو بخش تقسیم کرده اید. عمل تقسیم با عملگر (/) انجام می گیرد .

مثال :

```
    class Exercise
{
    static void Main()
    {
        // Initializing various variables when declaring them with the
same data type
        double value1 = 224.58, value2 = 1548.26;
        var result = value1 / value2;
        System.Console.Write(value1);
        System.Console.Write(" / ");
        System.Console.Write(value2);
        System.Console.Write(" = ");
        System.Console.WriteLine(result);
        System.Console.WriteLine();
    }
}

```

این نتیجه ی زیر را ارائه می دهد.

```
    224.58 / 1548.26 = 0.145053156446592
Press any key to continue…
```

هیچ گاه چیزی را به صفر تقسیم نکنید .

به کاربری عملگر تقسیم

- برای این منظور، فایل را به این صورت تغییر دهید .

```
    class Order
{
    static void Main()
```

```
    {
        const decimal priceOneShirt = 0.95M;
        const decimal priceAPairOfPants = 2.95M;
        const decimal priceOneDress = 4.55M;
        const decimal discountRate = 0.20M;// 20%
        const decimal taxRate = 5.75M; // 5.75%
        string customerName = "James Burreck", homePhone = "(202) 301-
7030";
        uint numberOfShirts = 5, numberOfPants = 2, numberOfDresses = 3;
        uint totalNumberOfItems;
        decimal subTotalShirts, subTotalPants, subTotalDresses;
        decimal DiscountAmount, totalOrder, netPrice, taxAmount,
salesTotal;
        decimal amountTended, difference;
        uint orderMonth = 3, orderDay = 15, orderYear = 2002;
        totalNumberOfItems = numberOfShirts + numberOfPants +
numberOfDresses;
        subTotalShirts = priceOneShirt * numberOfShirts;
        subTotalPants = priceAPairOfPants * numberOfPants;
        subTotalDresses = numberOfDresses * priceOneDress;
        totalOrder = subTotalShirts + subTotalPants + subTotalDresses;
        DiscountAmount = totalOrder * discountRate;
        netPrice = totalOrder - DiscountAmount;
        taxAmount = totalOrder * taxRate / 100;
        salesTotal = netPrice + taxAmount;
        amountTended = 50M;
        difference = amountTended - salesTotal;
        System.Console.WriteLine("-/- Georgetown Cleaning Services -/-");
        System.Console.WriteLine("====================================");
        System.Console.Write("Customer:  ");
        System.Console.WriteLine(customerName);
        System.Console.Write("Home Phone: ");
        System.Console.WriteLine(homePhone);
        System.Console.Write("Order Date: ");
        System.Console.Write(orderMonth);
        System.Console.Write('/');
        System.Console.Write(orderDay);
        System.Console.Write('/');
        System.Console.WriteLine(orderYear);
        System.Console.WriteLine("------------------------------------");
        System.Console.WriteLine("Item Type Qty Unit/Price Sub-Total");
        System.Console.WriteLine("------------------------------------");
        System.Console.Write("Shirts     ");
        System.Console.Write(numberOfShirts);
        System.Console.Write("    ");
        System.Console.Write(priceOneShirt);
        System.Console.Write("    ");
        System.Console.WriteLine(subTotalShirts);
        System.Console.Write("Pants      ");
        System.Console.Write(numberOfPants);
        System.Console.Write("    ");
        System.Console.Write(priceAPairOfPants);
        System.Console.Write("    ");
        System.Console.WriteLine(subTotalPants);
        System.Console.Write("Dresses    ");
        System.Console.Write(numberOfDresses);
        System.Console.Write("    ");
        System.Console.Write(priceOneDress);
        System.Console.Write("    ");
        System.Console.WriteLine(subTotalDresses);
        System.Console.WriteLine("------------------------------------");
```

```
53          System.Console.Write("Number of Items: ");
54          System.Console.WriteLine(totalNumberOfItems);
55          System.Console.Write("Total Order:    ");
56          System.Console.WriteLine(totalOrder);
57          System.Console.Write("Discount Rate:  ");
58          System.Console.Write(discountRate * 100);
59         System.Console.WriteLine('%');
60          System.Console.Write("Discount Amount: ");
61          System.Console.WriteLine(DiscountAmount);
62          System.Console.Write("After Discount: ");
63          System.Console.WriteLine(netPrice);
64         System.Console.Write("Tax Rate:       ");
65          System.Console.Write(taxRate);
66          System.Console.WriteLine('%');
67          System.Console.Write("Tax Amount:     ");
68          System.Console.WriteLine(taxAmount);
69          System.Console.Write("Net Price:      ");
70          System.Console.WriteLine(salesTotal);
71          System.Console.WriteLine("====================================");
72          System.Console.Write("Amount Tended:  ");
73          System.Console.WriteLine(amountTended);
74          System.Console.Write("Difference:     ");
75          System.Console.WriteLine(difference);
          System.Console.WriteLine("====================================");
          System.Console.ReadKey();
      }
}
```

- برنامه را اجرا کنید. نتیجه ی زیر را به دست می دهد .

```
1
2
3                -/- Georgetown Cleaning Services -/-
4==================================
5Customer:  James Burreck
6Home Phone: (202) 301-7030
7Order Date: 3/15/2002
8------------------------------------
9Item Type Qty Unit/Price Sub-Total
10------------------------------------
11Shirts     5    0.95    4.75
12Pants      2    2.95    5.90
13Dresses    3    4.55    13.65
14------------------------------------
15Number of Items: 10
16Total Order:    24.30
17Discount Rate:  20.00%
18Discount Amount: 4.8600
19After Discount: 19.4400
20Tax Rate:       5.75%
21Tax Amount:     1.39725
22Net Price:      20.83725
23==================================
Amount Tended:  50
Difference:     29.16275
==================================
```

- پنجره ی DOS را بسته و به محیط برنامه نویسی برگردید .

تقسیم مرکب (Compound division)

به خاطر دارید که می توان مقداری را به متغیری اضافه کرد، در آن ضرب کرد، یا از آن کسر کرد، سپس نتیجه ی آن را به خود متغیر اختصاص داد. این عملیات را می توان برای تقسیم نیز پیاده کرد. نمونه ی آن را در زیر مشاهده می کنید .

```
        class Exercise
{
    static void Main()
    {
        double value = 12.75;
        System.Console.Write("Value = ");
        System.Console.WriteLine(value);
        value = value / 2.42;
        System.Console.Write("Value = ");
        System.Console.WriteLine(value);
    }
}
```

نتیجه ی زیر حاصل می گردد .

```
1        Value = 12.75
2Value = 5.26859504132231
3Press any key to continue...
```

زبان C# برای این منظور عملگر /= را ارائه می دهد. مثال زیر این عملگر را به کار می برد .

```
?1
 2        class Exercise
 3{
 4    static void Main()
 5    {
 6        double value = 12.75;
 7        System.Console.Write("Value = ");
 8        System.Console.WriteLine(value);
 9        value /= 2.42;
10        System.Console.Write("Value = ");
11        System.Console.WriteLine(value);
12    }
13}
14
```

باقی مانده (Remainder)

مقدمه

عملیات تقسیم نتیجه ای با رقم اعشار در اختیار شما قرار می دهد، چنانچه عددی فرد (مثل ۱۴۷(در آن به کار ببرید. گاهی اوقات به باقی مانده ی عمل تقسیم نیاز داریم. برای مثال، تصور کنید ۲۶ بازیکن کودک در استادیوم حضور دارند، و بازی در شرف آغاز است. مطلع هستید که برای هر تیم به ۱۱ بازیکن نیاز است. اگر بازی با تعداد صحیح شروع شود، چند نفر بیرون از بازی منتظر می مانند؟

عملگر باقی مانده (remainder operation) که با علامت (%) نشان داده می شود، این عملیات را انجام می دهد. عملگر فوق با گرفتن5 + Shift ، فعال می شود .

مثال :

```
class Exercise
{
    static void Main()
    {
        var players = 18;
        var remainder = players % 11;
how many players will wait?..        // When the game starts
        System.Console.Write("Out of ");
        System.Console.Write(players);
");.        System.Console.Write(" players
        System.Console.Write(remainder);
        System.Console.WriteLine(" players will have to wait when the
game starts.\n");
    }
}
```

نتیجه ی زیر به دست می آید .

```
7 players will have to wait when the game .        Out of 18 players
starts.
Press any key to continue...
```

باقی مانده ی مرکب (compound remainder)

همان طور که در عملگرهای دیگر ریاضی مشاهده کردید، می توان پس از به دست آوردن باقی مانده ی یک متغیر نتیجه ی آن را دوباره به خود متغیر تخصیص داد .

مثال :

```
        class Exercise
{
    static void Main()
```

```
    {
        var players = 18;
how many players will wait?..        // When the game starts
        System.Console.Write("Out of ");
        System.Console.Write(players);
");.         System.Console.Write(" players
        players = players % 11;
        System.Console.Write(players);
        System.Console.WriteLine(" players will have to wait when the
game starts.\n");
    }
}
```

زبان C# با فراهم کردن عملگر %=، فرایند را سرعت می بخشد .

مثال :

```
      class Exercise
{
    static void Main()
    {
        var players = 18;
how many players will wait?..        // When the game starts
        System.Console.Write("Out of ");
        System.Console.Write(players);
");.         System.Console.Write(" players
        players %= 11;
        System.Console.Write(players);
        System.Console.WriteLine(" players will have to wait when the
game starts.\n");
    }
}
```

Bit Operations

مقدمه

همان طور که در دروس پیشین ذکر کردیم، کامپیوتر اطلاعات دریافتی را در قسمت های بسیار کوچکی به نام Bit در حافظه ی خود ذخیره می کند. هر Bit در خود یا ۰ دارد یا ۱. می توان Bit را با جا به جایی مقدار های ۰ و۱ (۰ به ۱ یا ۱ به ۰) دستکاری کرد .

عملیاتی که روی Bit انجام می گیرد، فقط با ۰ و ۱ سروکار دارند. به عبارت دیگر، هر عدد مبنای ۱۶ (hexadecimal) یا دهدهی (decimal) باید ابتدا به دودویی (binary) تبدیل شود .

عملیاتی که در برنامه نویسی Microsoft Windows (Win32) کاربرد فراوان دارد، OR operationاست که به آن خواهیم پرداخت .

معکوس کردن (" Bit" Reversing) " a " bit

به خاطر دارید که هرBit ، تنها یک مقدار دربردارد : یا ۰ و یا ۱. وارونه سازی (معکوس کردن) جزئی از عملیات ایجاد تغییر در Bitمی باشد. برای مثال، اگر در یک Bit مقدار ۱ را دارید، به ۰ تبدیل می شود و اگر مقدار ۰ را دارید به ۱ تبدیل می شود. زبان C# با عملگر ~، این عملیات را ممکن می سازد .

برای نمونه رقم ۲۸۶ را در نظر بگیرید. این رقم (decimal)دهدهی است با سیستم دودویی (binary) به این شکل در می آید : ۱۰۰۰۱۱۱۱۰ .حال، می توان هر bit را به صورت زیر وارونه کرد .

برای استفاده ی صحیح از عملگرnot ، ~را در سمت چپ مقدار قرار دهید .

```
1
2       class Exercise
3{
4   static void Main()
5   {
6       var number1 = 286;
7       System.Console.Write("286 = ");
8       System.Console.WriteLine(number1);
9       System.Console.Write("Not 286 = ");
10      System.Console.WriteLine(~number1);
11  }
}
```

نتیجه ای که حاصل می شود .

```
1286 = 286
2Not 286 = -287
3Press any key to continue…
```

پیوستگی بیتی (Bitwise Conjunction)

عبارت است از پیوند دادن (یا ضمیمه کردن) محتوای یک bit به bit دیگر. برای این منظور، C#عملگر &را تعبیه کرده .

برای پیوند دادن محتوای دوbit ، به خاطر داشته باشید که اول باید آن ها را به سیستم دودویی (binary) تبدیل کنید .

چنانچه bit ای با مقدار ۰ به bit دیگری با همین مقدار اضافه شود، نتیجه ۰ خواهد بود .

Bit0	0
Bit1	0
Bit0 And Bit1	0

چنانچه bit با مقدار ۱، به bit ای با مقدار ۰ اضافه شود، بازهم نتیجه ۰ خواهد بود .

Bit0	1
Bit1	0
Bit0 And Bit1	0

اگر bit با مقدار ۰، به bit ای با مقدار ۱ اضافه شود، نتیجه ۰ خواهد بود .

Bit0	0
Bit1	1
Bit0 And Bit1	0

اگرچنانچه bit ای با مقدار ۱ به bit دیگری با همین مقدار اضافه شود، نتیجه ۱ خواهد شد .

Bit0	1
Bit1	1
Bit0 And Bit1	1

تصور کنید می خواهید عدد ۲۸۶ bit را به ۴۷۵ اضافه کنید .رقم دهدهی ۲۸۶ را که به سیستم دودویی تبدیل کنید، این نتیجه به دست می آید.100011110 : نسخه ی دودویی رقم ۴۰۷۵ (که دهدهی می باشد) معادل 111111101011 : می باشد. بر اساس چهار قاعده ی فوق، می توان این دو رقم را به صورت زیر جمع کرد .

286	0	0	0	1	0	0	0	1	1	1	1	0
4075&286	1	1	1	1	1	1	1	0	1	0	1	1
286 &4075	0	0	0	1	0	0	0	0	1	0	1	0

بنابراین، ۳۰۵ ^ ۲۸۵۳ نتیجه می دهد : ۱۰۱۱۱۰۰۱۱۱۱۰ که برابر است با

	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	256	128	64	32	16	8	4	2	1
286 &4075	1	0	0	0	0	1	0	1	0
	256	0	0	0	0	8	0	2	0

یعنی ؛ ۲۸۶ ^ ۴۰۷۵ = ۲۰۴۸ + ۵۱۲ + ۲۵۶ + ۱۲۸ + ۱۶ + ۸ + ۴ + ۲ = ۲۹۷۴1

البته، عمل فوق را به این صورت نیز می توان محاسبه کرد .

```
1                class Exercise
2{
    static void Main()
3   {
```

```
        var number1 = 618;
        var number2 = 2548;
        var result = number1 ^ number2;
        System.Console.WriteLine(result);
    }
}
```

نتیجه می دهد .

```
2974
Press any key to continue…
```

می توان شماری bit را به صورتی که گفته شد، از متغیری خارج کرد و بعد نتیجه ی حاصل را برای خود متغیر به کاربرد .

مثال :

```
class Exercise
{
    static void Main()
    {
        var number = 618;
        System.Console.WriteLine(number);
        number = number ^ 38;
        System.Console.WriteLine(number);
    }
}
```

نتیجه می دهد .

```
618
588
Press any key to continue...
```

همچنین می توان با استفاده از عملگر ^= به همین نتیجه دست یافت .

```
class Exercise
{
    static void Main()
    {
        var number = 618;
        System.Console.WriteLine(number);
        number ^= 38;
        System.Console.WriteLine(number);
    }
}
```

انتقال Bit ها از راست به چپ

عبارت است از جا به جایی bit ها از سمت راست به سمت چپ. در این مورد نیز، لازم است پیش از انجام عملیات عدد به معادل دودویی خود تبدیل شود C# .این عملیات را با عملگر >>پشتیبانی می کند .

رقم ۷۴۱ را در نظر بگیرید. معادل دودویی آن برابر است با : ۱۰۱۱۱۰۰۱۰۱ که به صورت زیر نمایش داده می شود .

1	0	1	1	1	0	0	1	0	1

به منظور انجام عملیات فوق، باید تک تک مقدارها را (بسته به تعداد دفعات جا به جایی) از سمت راست به چپ جا به جا کرد. با انتقال یک bitاز چپ به راست، موقعیت تنها یک bit تغییر می کند. برای مثال، bitای در موقعیت x قرار می گیرد و bitدیگری سمت چپ آن در موقعیت .y مقدار bit x را جایگزین مقدار bit y می کنیم .در صورتی که bit x در راست ترین موقعیت قرار داشت، مقدار ۰ دریافت می کند .

Original		1	0	1	1	1	0	0	1	0	1
<<by 1	1	0	1	1	1	0	0	1	0	1	0

نتیجه ی ۱۰۱۱۱۰۰۱۰۱۰ را به دست می دهد. نتیجه ی دهدهی (decimal) آن به این صورت محاسبه شده .

	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	1024	512	256	128	64	32	16	8	4	2	1
741 <<1	1	0	1	1	1	0	0	1	0	1	0
	1024	0	256	128	64	0	0	8	0	2	0

همچنین، این نتیجه حاصل می شود .

```
741 <<1 =1024 + 256 + 128 + 64 + 8 + 2=1482
```

همان طور که از مثال بالا پیدا است، باید برای پیاده سازی این عملیات از عملگر >>استفاده کرد .

```
    class Exercise
{
static void Main()
{
    System.Console.WriteLine(741 << 1);
 }
 }
```

```
6
7
```

نتیجه :

```
1        1482
2Press any key to continue...
```

به همین ترتیب، می توان bit ها را بیش از یک واحد به سمت چپ انتقال داد .

```
1
2        class Exercise
3{
4    static void Main()
5    {
6        var number = 248 << 5; System.Console.WriteLine(number);
7                               }
                                }
```

نتیجه

```
1                    7936
2Press any key to continue…
```

بنابراین، می توان چند bit را به سمت چپ (یک متغیر) جا به جا کرد، سپس نتیجه ی حاصله را برای خود متغیر اعمال کرد .

```
1
2        class Exercise
3{
4    static void Main()
5    {
6        var number = 248;
7        System.Console.WriteLine(number);
8        number = number << 5; System.Console.WriteLine(number);
9                             }
                              }
```

نتیجه ی زیر را به دست می دهد.

```
1        248
27936
3Press any key to continue...
```

#Cعملیات فوق را با به کاربردن عملگر =>>برای کاربر بسیار آسان می کند.

```
1
2           class Exercise
3{
4       static void Main()
5       {
6           var number = 248;
7           System.Console.WriteLine(number);
8           number <<= 5;
9          System.Console.WriteLine(number);
10      }
}
```

انتقال Bit ها به سمت راست

می توان bit ها را در سمت راست قرار داد. برای این منظور تمام مراحلی که برای انتقال bit ها به چپ انجام می دادید را به صورت معکوس برای این فرایند پیاده کنید. زبان C# با عملگر <<، این پروسه را امکان پذیر می کند.