第64章 演算子オーバーロード その3

今回は、単項演算子と関係演算子のオーバーロードについてやります。

単項演算子のオーバーロードの仕方は簡単です。

```
public static データ型 operator 演算子(データ型 オペランド){...}
```

関係演算子の場合、戻り値のデータ型はbool型となります。また、「<」をオーバーロードしたら「>」もオーバーロードしなくてはなりません。つまり、対になっている演算子は、両方オーバーロードする必要があります。

また、「==」や「!=」をオーバーロードするときは、EqualsとGetHashCodeメソッドをオーバーライドしなくてはなりません。(しないとコンパイラが警告を発します。)

Equalsはまあ、よいのですがGetHashCodeはなかなかくせものです。GetHashメソッドについては後の章で解説することにして、この章では

```
return base.GetHashCode();
```

としています。baseは、基本クラスのメソッドを呼び出すときに使います。

そもそもEqualsとかGetHashメソッドなどは、どこから来ているのかというとSystem.Objectクラスから来ています。すべてのクラスは暗黙のうちにObjectクラスから派生しています。.net型でいうとobject型です。この型には、どんな型でも代入することができます。これに関しても後に解説します。

では、サンプルを見てみましょう。

```
// opover03.cs

using System;

class MyPosition
{
    int nX, nY;

    public int x
    {
        get
        {
            return nX;
        }
        set
        {
            nX = value;
        }
    }
    public int y
    {
        get
        {
            return nY;
        }
        set
        {
            nY = value;
        }
    }

    public static MyPosition operator +(MyPosition a, MyPosition b)
    {
        MyPosition c = new MyPosition();

        c.nX = a.nX + b.nX;
        c.nY = a.nY + b.nY;

        return c;
    }

    public static MyPosition operator -(MyPosition a, MyPosition b)
    {
        MyPosition c = new MyPosition();

        c.nX = a.nX - b.nX;
        c.nY = a.nY - b.nY;

        return c;
```

```
}

public static MyPosition operator *(MyPosition a, int b)
{
    MyPosition c = new MyPosition();
    c.x = a.x * b;
    c.y = a.y * b;
    return c;
}

public static MyPosition operator *(int b, MyPosition a)
{
    MyPosition c = new MyPosition();
    return a * b;
}

public static MyPosition operator +(MyPosition a)
{
    return a;
}

public static MyPosition operator -(MyPosition a)
{
    return a * -1;
}

public static bool operator ==(MyPosition a, MyPosition b)
{
    if (a.x == b.x && a.y == b.y)
        return true;
    else
        return false;
}

public static bool operator !=(MyPosition a, MyPosition b)
{
    if (a == b)
        return false;
    else
        return true;
}

public override bool Equals(object obj)
{
    MyPosition a;
    if (this.GetType() != obj.GetType())
        return false;
    a = (MyPosition)obj;
    if (a.x == this.x && a.y == this.y)
        return true;
    else
        return false;
}

public override int GetHashCode()
{
    return base.GetHashCode();
}

public override string ToString()
{
    return string.Format("({0}, {1})", this.x, this.y);
}

public MyPosition()
{
    nX = 0;
    nY = 0;
}

public MyPosition(int m, int n)
{
    nX = m;
    nY = n;
```

```
    }
}

class opover01
{
    public static void Main()
    {
        MyPosition A = new MyPosition(3, 5);
        MyPosition B = new MyPosition(4, 6);
        MyPosition C = new MyPosition(7, 11);
        MyPosition D = new MyPosition(-1, -1);

        if (C == (A + B))
            Console.WriteLine("C = A + Bです");
        else
            Console.WriteLine("C = A + Bではありません");

        if (D != (A - B))
            Console.WriteLine("D = A - Bではありません");
        else
            Console.WriteLine("D = A - Bです");

        if (C.Equals(A + B))
            Console.WriteLine("C = A + Bです");
        else
            Console.WriteLine("C = A - Bではありません");

        Console.WriteLine("C = {0}", C.ToString());
    }
}
```

単項演算子+は、

```
public static MyPosition operator +(MyPosition a)
{
    return a;
}
```

のようにオーバーロードしました。aに対して何の作用もしませんね。

これに対して、単項演算子-は、

```
public static MyPosition operator -(MyPosition a)
{
    return a * -1;
}
```

のように、すでに定義してある*演算子を利用しました。

関係演算子の==は、

```
public static bool operator ==(MyPosition a, MyPosition b)
{
    if (a.x == b.x && a.y == b.y)
        return true;
    else
        return false;
}
```

フィールドnXとnYが両方等しい(ここでは、プロパティのx,yを利用)時のみa,bが等しく、それ以外は、等しくないと定義しました。!=演算子の定義については、==演算子の定義を利用しています。

さて、Equalsメソッドのオーバーロードですが、引数がobjct型です。そこで、このobject型に、どんな型が代入されているかを調べます。この型がMyPosition型と異なれば、直ちにfalseを返すようにします。これには、GetTypeメソッド(Objectクラスから引き継いできている)を使います。

```
public Type GetType ()
```

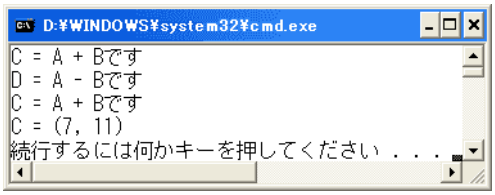
現在のインスタンスの型を表すTypeインスタンスを返します。具体的には次のようにしていますね。

```
public override bool Equals(object obj)
{
    MyPosition a;
    if (this.GetType() != obj.GetType())
        return false;
    a = (MyPosition)obj;
    if (a.x == this.x && a.y == this.y)
        return true;
```

```
    else
        return false;
}
```

ついでに、ToStringメソッドもオーバーライドしています。MyPositionインスタンスの nX, nYフィールドを使って「(nX, nY)」の形の文字列を返すようにしています。

では、実行結果を見てみましょう。



[C# Index] [総合Index] [Previous Chapter] [Next Chapter]

Update 09/Oct/2006 By Y.Kumei