第62章 演算子オーバーロード その1

自分でクラスを作成したら、演算子に独自の定義を設けることができます。これを演算子のオーバーロードといいます。

演算子は、オペランドの数により単項演算子(例-Aのマイナス)、2項演算子(A + Bのプラス)などに分類することができます。この章では2項演算子の+と-をオーバーロードしてみます。

演算子をオーバーロードできるのは自分で作ったクラスのみです。

```
public static 演算後の型 operator 演算子(データ型 オペランド1, データ型 オペランド){.....}
```

オペランドのうち少なくとも一つは自分のクラス型でなくてはなりません。また、引数にoutやrefは使えません(outやrefについて忘れてしまった人は第22章を見てください)。

また、publicとstaticがついていることに注意してください。

この章では、サンプルとして2次元ベクトルもどきを使います。

(「もどき」とあるのは、数学的厳密さが全然ないからです。)

ベクトルは(2,5)のように2つのint型の数値の組み合わせで表現するものとします。

(2,5)と(5,2)は、異なるものとします。

さて、このベクトルもどき空間で+と-の演算を定義します。

(a,b) + (c,d)は(a+c, b+d)とします。

(a,b) - (c,d)は(a-c, b-d)とします。

2つのベクトル(a,b)と(c,d)が等しいのは、a=cかつb=dの時としますが、この章では 定義していません。暗黙のうちに等しいとします。

なんだか、いい加減な定義ですが、試しにプログラムを作ってみましょう。

MyPositionクラスがベクトルもどきを表現するクラスです。(2つのint型の組み合わせは 位置を表すとも考えられるのでこの名称を使いました。)

```
// opover01.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 MyPosition(int m, int n)
    {
        nX = m;
        nY = n;
    }

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

class opover01
{
    public static void Main()
    {
        MyPosition mp = new MyPosition(3, 5);
        MyPosition np = new MyPosition(7, 3);

        MyPosition A, B;

        A = mp + np;
        Console.WriteLine("A = ({0}, {1})", A.x, A.y);

        B = mp - np;
        Console.WriteLine("B = ({0}, {1})", B.x, B.y);

    }
}
```

ベクトルを表すクラスMyPositionには、インスタンスフィールドとしてnX,nYを持っています。これが、(a,b)のa,bに相当するものと考えてください。

この、nX,nYに値を代入したり、読み出したりするx,yプロパティがあります。

そして、

```
public static MyPosition operator +(MyPosition a, MyPosition b){...}
public static MyPosition operator -(MyPosition a, MyPosition b){...}
```

というように、演算子をオーバーロードしています。

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

と、いうようにフィールドを使って演算子の定義をしていますが、もちろんプロパティを使って

```
c.x = a.x + b.x;
```

のように書いても問題ありません。

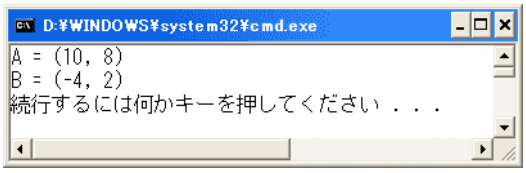
また、引数有りと無しのコンストラクタも用意しておきました。

Mainメソッドでは、

```
MyPosition mp = new MyPosition(3, 5);
MyPosition np = new MyPosition(7, 3);
```

のように、mpとnpというベクトルを生成しています。この2つのベクトルの 足し算、引き算をやっています。加算の結果は(3+7, 5+3)で(10,8)となります。減算の結果は(3-7,5-3)で(-4,2)となります。

実行結果は次のようになります。



Update 07/Oct/2006 By Y.Kumei