

第15章 for文



for文は繰り返し処理を行います。非常によく使う構文です。

for文の説明に入る前に、少し演算子の追加説明があります。

```
int a = 10;
a = a + 1;
```

とあった場合、aは11となりますね。これを次のように書くことができます。

```
int a = 10;
a++;
```

「++」のことをインクリメント演算子と呼びます。

同様に

```
int a = 10;
a--;
```

とあれば、a--は、a = a - 1の意味です。「--」をデクリメント演算子といいます。

「++」や「--」は、オペランドの前に置くこともできます。後ろに置いた場合は、その式が次に評価されるときに1増減され、前に置いたときはすぐに評価されます。

```
// increment01.cs

using System;

class incremet01
{
    public static void Main()
    {
        int a = 10;

        Console.WriteLine("a = {0}", a++);
        Console.WriteLine("a = {0}", a);

        Console.WriteLine("-----");

        a = 10;
        Console.WriteLine("a = {0}", ++a);
        Console.WriteLine("a = {0}", a);
    }
}
```

最初の

```
Console.WriteLine("a = {0}", a++);
```

では、a++は10のままで、次に評価されるとき11となります。

```
Console.WriteLine("a = {0}", ++a);
```

では、この時すでにインクリメントされているので++aの値は11となっています。

では、実行結果を見てみましょう。

```
D:\WINDOWS\system32\cmd.exe
a = 10
a = 11
-----
a = 11
a = 11
続行するには何かキーを押してください . . .
```

さて、for文に戻りますが、これは次のように書きます。

```
for (初期化子; 論理式; 反復子)
{
    ...
}
```

初期化子は、変数の初期化を行います。この変数の値により反復を続けるかやめるかの基準となります。

論理式がtrueの間、...が繰り返し実行されます。

...が1回実行されるたびに反復子により、変数の値を変化させます。通常「++」か「--」を使います。

```
for (i = 0; i < 10; i++)
{
    ...
}
```

とすれば、iに最初に0が設定されます。そして、...が実行されるとi++によりiの値が1増加します。iが10になると、反復処理が停止します。

なお、...が1つの文のみの時は、{}を省略できます。

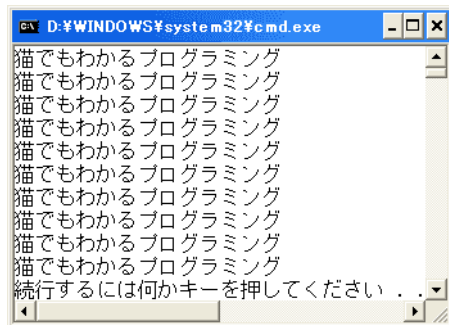
```
// for01.cs

using System;

class for01
{
    public static void Main()
    {
        int i;

        for (i = 0; i < 10; i++)
            Console.WriteLine("猫でもわかるプログラミング");
    }
}
```

実行結果は、次のようになります。「猫でもわかる...」が10回表示されますね。



さて、この例では変数iはforループの外で宣言しています。

forループ以外では、この変数を使わないとわかっている場合は

```
for (int i = 0; i < 10; i++)
{
    ...
}
```

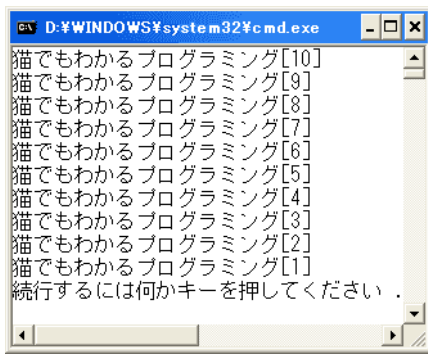
と書くことができます。変数iはループ外では見えないので使用できません。また、ループ外で新規にint iと宣言することもできません。

```
// for02.cs

using System;

class for02
{
    public static void Main()
    {
        for (int i = 10; i > 0; i--)
            Console.WriteLine("猫でもわかるプログラミング [{0}]", i);
    }
}
```

この例ではfor文の中で宣言しています。そして、iは初期値として10が代入されます。ループを回るたびにiの値は1ずつ減少します。iが0になったら終了です。



さて、このプログラムで

```
for (int i = 10; i > 0; --i)
{
    ...
}
```

とするとどうなるのでしょうか。

「こりゃ、きっと実行回数が1回減るんじゃないか・・・」

はい、残念でした。結果としては同じです。

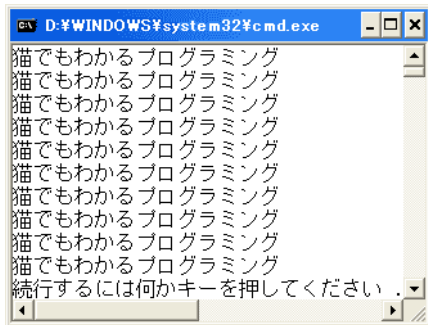
最初は初期値がそのまま使われます。...が実行されると--が実行されます。この時の値はまだ減らされていません。次にi > 0でiの値が評価されるのでここで1減らされます。従ってi--としても--としても同じなのです。実験してみましょう。

// for03.cs

```
using System;
```

```
class for03
{
    public static void Main()
    {
        for (int i = 10; i > 0; --i)
            Console.WriteLine("猫でもわかるプログラミング");
    }
}
```

結果は次のようになります。(ループ内での評価を行わないようにしています)



やはり、「猫でもわかる・・・」は10回表示されますね。

さて、for(...; ...; ...)の項目は常に全部書かなくてはいけない、ということはありません。自由に省略ができます。この場合反復中止の目印となる変数は自前で調整する必要があります。

// for04.cs

```
using System;
```

```
class for04
{
    public static void Main()
    {
        int i = 0;

        for( ; i < 5; )
        {
            Console.WriteLine("猫でもわかるプログラミング [{0}]", i);
            i++;
        }
    }
}
```

```

Console.WriteLine("-----");

for (i = 0; ; i++)
{
    if (i >= 5)
        break;
    Console.WriteLine("猫でもわかるプログラミング [{0}]", i);
}

Console.WriteLine("-----");

i = 0;
for ( ; ; )
{
    Console.WriteLine("猫でもわかるプログラミング [{0}]", i);
    i++;
    if (i >= 5)
        break;

}

}
}

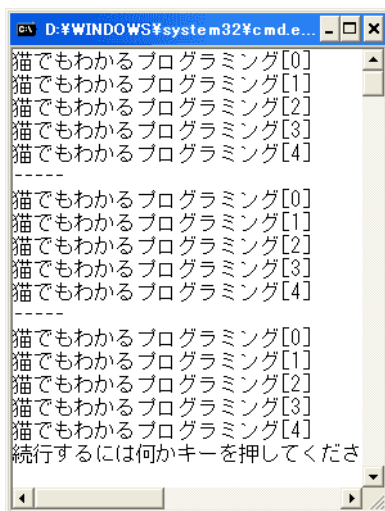
```

最初の例では、反復子が記述されていないので、自前で*i*の値をインクリメントしています。

次の例では、論理式が無いので、自前で*i*の値を監視して5以上になった場合はbreak文でループを脱出しています。

3番目の例では、全部自前で処理しています。特に、3番目の例は無限ループとして利用できます。(何らかの方法でループを脱出しないと、スタックを使い果たしてしまいます)

実行結果はどれも同じですね。



```

D:\WINDOWS\system32\cmd.e...
猫でもわかる プログラミング[0]
猫でもわかる プログラミング[1]
猫でもわかる プログラミング[2]
猫でもわかる プログラミング[3]
猫でもわかる プログラミング[4]
-----
猫でもわかる プログラミング[0]
猫でもわかる プログラミング[1]
猫でもわかる プログラミング[2]
猫でもわかる プログラミング[3]
猫でもわかる プログラミング[4]
-----
猫でもわかる プログラミング[0]
猫でもわかる プログラミング[1]
猫でもわかる プログラミング[2]
猫でもわかる プログラミング[3]
猫でもわかる プログラミング[4]
続行するには何かキーを押してください

```

無限ループを使えると、いろいろ面白いプログラムを作ることができます。少しずつ解説していきます。

[\[C# Index\]](#) [\[総合Index\]](#) [\[Previous Chapter\]](#) [\[Next Chapter\]](#)

Update 22/Aug/2006 By Y.Kumei

当ホーム・ページの一部または全部を無断で複写、複製、転載あるいはコンピュータ等のファイルに保存することを禁じます。