

第17章 foreach文



C/C++では、あまりなじみがありません(C++のアルゴリズム関数にfor_eachメソッドというのがありますが...)がVBではおなじみのforeach文がC#では使えます。

foreach文は次のような形になります。

```
foreach (データ型 変数名 in 配列名)
{
    ...
}
```

foreach文では、添え字のもっとも小さいものから順番に調べていき、変数に代入されます。

...が1つの文だけであれば{}は省略できます。

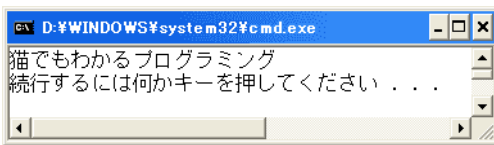
```
// foreach01.cs

using System;

class foreach01
{
    public static void Main()
    {
        string[] st = new string[] { "猫でも", "わかる", "プログラミング" };

        foreach (string s in st)
            Console.Write(s);
        Console.WriteLine();
    }
}
```

この例では、string型の配列stに対してforeach文が適応されています。st[0]から順に変数sに代入され、Console.Write(s)で画面に表示されます。結局画面には「猫でもわかるプログラミング」と表示されるはずですが、



次は、もうちょっと実用的な例です。

その前に、ArrayListクラスについて、ちょっと説明しておきます。

第6章で解説した配列について不満はありませんか。ここで解説した配列は、最初から要素数が決まっていた。途中で配列の要素数を加減できませんでした。ユーザー入力により、自由に配列の長さが伸張すれば便利ですね。それを実現するのがArrayListクラスです。(ただし、C#2.0の出現によりもっと効率的なものが出てきました。ジェネリックListクラスというものです。ジェネリックについては後の章で解説します。)

ArrayListクラスは、サイズが動的に増加する配列を実現します。

System.Collections名前空間で定義されてるので、using System.Collections;をプログラムの冒頭に書いておくと便利です。

さしあたって必要なメソッドはAddメソッドです。

```
public virtual int Add (
    Object value
)
```

配列の末尾にデータ(value)を追加します。

Countプロパティは要素の数を返します。

```
ArrayList インスタンス名 = new ArrayList();
```

で、オブジェクトを生成してから使います。

```
// foreach02.cs

using System;
using System.Collections;

class foeach02
{
    public static void Main()
    {
        ArrayList al = new ArrayList();
```

```

while (true)
{
    Console.Write("(xで終了)データ:");

    string strData = Console.ReadLine();

    if (strData == "")
        return;

    if (!char.IsDigit(strData[0]) && strData[0] != '-')
        break;

    double dData = double.Parse(strData);
    al.Add(dData);
}

double sum = 0.0;

if (al.Count == 0)
{
    Console.WriteLine("データが有りません");
    return;
}

foreach (double d in al)
{
    sum += d;
}

Console.WriteLine("データ数:{0, 6}¥n合計:{1, 10}¥n平均:{2, 10}",
    al.Count, sum, sum / al.Count);
}
}

```

まず、ArrayListクラスのインスタンスを作成しておきます。

無限ループで、ユーザーにデータを入力させます。

ユーザーが何も入力せずにいきなり、エンターキーを押したときは、

```

if (strData == "")
    return;

```

で、プログラムを終了してしまいます。

入力された文字列を解析して、最初の文字が10進数でも、マイナス記号でも無いときはbreak文で無限ループを脱出します。

```

if (!char.IsDigit(strData[0]) && strData[0] != '-')
    break;

```

charクラスのIsDigitプロパティについては、[第12章](#)で解説がありますので忘れた人は読んでみてください。

ここまで、進んでこれたら文字列をdouble型に変換してdDataに格納します。

(本当はユーザー入力をもう少しと検査しないと、実行時エラーが出ることがありますが、煩雑になるので省略しています。たとえば2pなど2文字目以降に数字以外が入力されたときなど)

double型に変換したデータを

```

al.Add(dData);

```

で、配列の末尾に追加しています。

ユーザーがいきなり、xとか数字以外を入力してしまったときは、配列には何も格納されていないので「データが有りません」と表示してプログラムを終了させます。

データが存在する場合は、foreach文で各要素の合計を求めます。

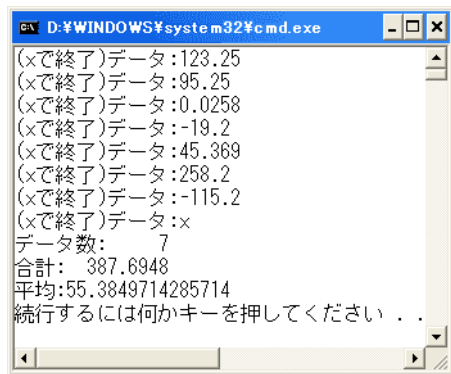
```

foreach (double d in al)
{
    sum += d;
}

```

これは、簡単ですね。alの添え字の最小値(0)から最大値までが順にdに入力され、sum += d;で合計が計算されます。もちろんfor文を使って合計を計算することも可能です。

合計が出たら、平均値を出すのは簡単ですね。



```
D:\WINDOWS\system32\cmd.exe
(xで終了)データ:123.25
(xで終了)データ:95.25
(xで終了)データ:0.0258
(xで終了)データ:-19.2
(xで終了)データ:45.369
(xで終了)データ:258.2
(xで終了)データ:-115.2
(xで終了)データ:x
データ数: 7
合計: 387.6948
平均: 55.3849714285714
続行するには何かキーを押してください . . .
```

データ入力を終了するには'x'に限らず、数字以外を入力してエンターキーを押せばよいですね。

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

Update 24/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。