第5章 C#のデータ型

プログラムで扱うデータは、その特性によりいずれかのデータ型に属します。

では、C#のデータ型にはどのようなものがあるのでしょうか。C/C++のデータ型に似ていますが、全然違う側面も持っています。

C#で取り扱うデータ型は大きく、「値型(value type)」と「参照型(reference type)」に分かれます。

データ型	値型(value type)
	参照型(reference type)

値型は、変数(valiable)を宣言すると、その変数はメモリ上に領域が確保されます。これに、値を代入すると(初期化すると)、確保された領域にその値が設定されます。

参照型の変数もやはり、メモリ上にその領域が確保されます。変数を初期化するとき、初期値は、ヒープ領域に確保され、変数が確保している領域にはその参照が代入されます。

「参照」とは、メモリのアドレスのことです。

値型はstructキーワードで宣言されています。(構造体です。C/C++の構造体とは全く異なります。また、列挙型をのぞきます。)

参照型はclassキーワードで宣言されています。

この他に、C/C++のポインタ(Pointer type)も扱えます。

さて、この章では比較的取り扱いが簡単な値型の中でも数値を取り扱う数値型について解説します。その他の型についてはもう少し後で解説を行います。数値型は、整数型と実数型に大別されます。

			C#型	ビット長	.NET型
数値型	整数型	符号付き	sbyte	8ビット	System.SByte
			short	16ビット	System.Int16
			int	32ビット	System.Int32
			long	64ビット	System.Int64
		符号なし	byte	8ビット	System.Byte
			ushort	16ビット	System.UInt16
			uint	32ビット	System.UInt32
			ulong	64ビット	System.UInt64
	実数型	浮動小数点数型	float	32ビット	System.Single
			double	64ビット	System.Double
		10進データ型	decimal	128ビット	System.Decimal

C#の型には.NET型というエイリアスがあります。これは、.NET Framework上で動作する他の言語と型を一致させるためのものです。

C#でプログラミングをする際、どちらの型を使ってもかまいません。また、プログラムの冒頭でusing System;とあれば、Systemを省略できます(System.Int32はInt32など)。また、これらの型はstructで宣言されているので、次のような静的フィールドを利用できます。(フィールドについては後の章で解説)

MaxValue, MinValue

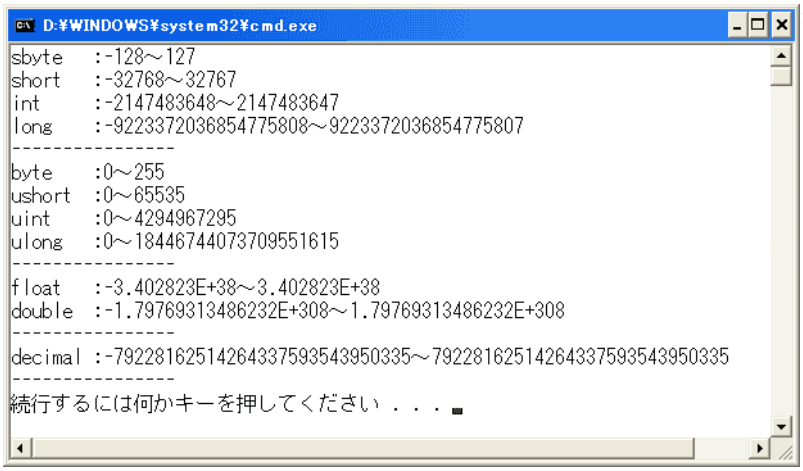
使い方は至って簡単で、Int32.MaxValueが、Int32型の最大値を表します。もちろんint.MaxValueとしても同じです。

```
// datatype01.cs

using System;

class datatype01
{
    public static void Main()
    {
        string format = "{0, -8}:{1}～{2}", dot = "----------------";

        Console.WriteLine(format, "sbyte",sbyte.MinValue, sbyte.MaxValue);
        Console.WriteLine(format, "short", short.MinValue, short.MaxValue);
        Console.WriteLine(format, "int", int.MinValue, int.MaxValue);
        Console.WriteLine(format, "long", long.MinValue, long.MaxValue);
        Console.WriteLine(dot);
        Console.WriteLine(format, "byte", byte.MinValue, byte.MaxValue);
        Console.WriteLine(format, "ushort", ushort.MinValue, ushort.MaxValue);
        Console.WriteLine(format, "uint", uint.MinValue, uint.MaxValue);
        Console.WriteLine(format, "ulong", ulong.MinValue, ulong.MaxValue);
        Console.WriteLine(dot);
        Console.WriteLine(format, "float", float.MinValue, float.MaxValue);
        Console.WriteLine(format, "double", double.MinValue, double.MaxValue);
        Console.WriteLine(dot);
        Console.WriteLine(format, "decimal", decimal.MinValue, decimal.MaxValue);
        Console.WriteLine(dot);
```

```
    }
}
```

書式制御文字列で{0, -8}とあるのは、インデックス0の引数のスペースを8つ分取って、左詰で表示しなさいという意味です。{0, 8}なら右詰になります。

結果は、次のようになります。

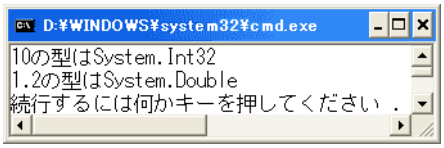


xxxE+zzというのは、xxxかける10のzz乗という意味です。

さて、整数の10はint型の範囲にも入りますし、byte型、sbyty型、short, ushort, uint, long, ulong, float, double, decimal型の範囲にも属します。型を調べるGetTypeメソッドというのがあります。これは、どんなオブジェクトにでも使えます。

```
// literal01.cs

using System;

class literal01
{
    public static void Main()
    {
        string format = "{0}の型は{1}";

        Console.WriteLine(format, 10, 10.GetType());
        Console.WriteLine(format, 1.2, (1.2).GetType());

    }
}
```

実行結果は次のようになります。



これを見てわかるように単なる整数リテラルはint型(int型におさまらないときは、uint型に、それにもおさまらないときはlong, まだだめなときはulong型)とみなされます。

1.54のような小数点数はdouble型とみなされます。

では、次のような場合はどうなるのでしょうか。

```
byte x = 10;
```

リテラル10はint型です。int型より小さなbyte型に代入できるのでしょうか。たまたま、int型の中でも小さな数字10なのでbyte型に代入してもよいのでしょうか。この例の場合は、コンパイラは文句を言いません。

```
int x = 10;
byte b;
b = x;
```

この例では、コンパイルエラーとなります。代入しても大丈夫だとわかっているときは 型キャストをします。型キャストは

(データ型)式

で行います。

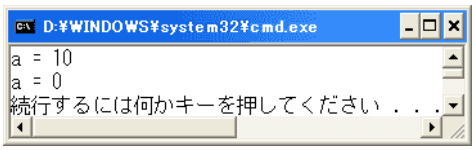
```
int x = 10;
byte b;
b = (byte)x;
```

とすれば、コンパイラは文句を言いません。しかし、型キャストをするときは慎重に行わないと、非常にわかりにくいバグが発生することがあります。

```
// cast01.cs

using System;
```

```
class cast01
{
    public static void Main()
    {
        byte a;
        int b = 10;
        a = (byte)b;

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

        // 256はbyte型の範囲を超え桁あふれが起る
        b = 256;
        a = (byte)b;
        Console.WriteLine("a = {0}", a);
    }
}
```

このプログラムをコンパイルしても、「警告0,エラー0」で通ってしまいます。しかし、実行すると



2回目のaの値が変ですね。これは、byte型に無理矢理255を代入したため桁あふれが起きたためです。あふれた桁は無視されるので0になってしまいます。

数値リテラルの型を明示的に示すこともできます。これには接尾辞をつけます。

分類	接尾辞	データ型
整数型	なし	int
	L(またはl)	long
	U(またはu)	uint
	UL(またはul)	ulong
実数型	F(またはf)	float
	D(またはd)	double
	M(またはm)	decimal

10Lと書くとこれは、long型の10という意味になります。

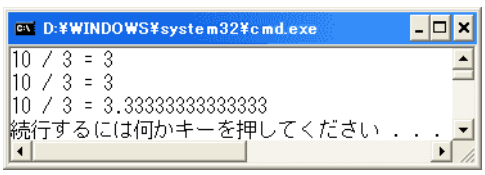
小数点数リテラルをdecimal型の変数に値を代入するときは必ずM(またはm)をつけないとエラーとなります。

二項式の内部ではデータ化を混在させることが可能です。この場合小さい型のは、大きい型に変換されます。(decimalだけは例外的で、1つがdecimalの時、他方がfloat,doubleである場合はエラーとなります。)

たとえば、int型 + byte型の時は、両方がint型とみなされます。

```
// cast02.cs

using System;

class cast02
{
    public static void Main()
    {
        int x = 10, y = 3;
        double z;

        z = x / y;
        Console.WriteLine("{0} / {1} = {2}", x, y, z);

        z = (double)(x / y);
        Console.WriteLine("{0} / {1} = {2}", x, y, z);

        z = ((double)x) / y;
        Console.WriteLine("{0} / {1} = {2}", x, y, z);
    }
}
```

実行結果は次のようになります。



x / yは、xもyもint型なのでx / yもint型です。この時x / y (=3.3333...)は切り捨てられ3となります。これが、double型に変換されzに代入されます。

第2の例では、(x / y)はint型でやはり、切り捨てられ3となり、これがdouble型にキャストされzに代入されます。

第3の例では、(double)x / yは型の大きい方に、合わせるのでyもdouble型となります。全体としてはdouble型で、その値は3.333...となります。これがzに代入されます。

Update 07/Aug/2006 By Y.Kumei