シフト演算

ビットシフト

通常私たちは使いなれている10進数を使ってプログラムをしていますが
機械語レベルでは全てが2進数であり、数値も命令コードもデータも例外ではありません

そこで、算術演算以外にビットを直接扱って演算することもあります
とくにビットを左右に移動させる**シフト演算**は時に非常に効果的です

シフト演算には**右シフト**と**左シフト**があります
右シフトは連続したビットを右に、左シフトはビットを左にずらします

0010 0110 -右に1シフト-> 0001 0011
0011 0101 -左に2シフト-> 1101 0100

例えばこのように2進ビット列を左右に振ることができます
これは8ビット型のデータをシフトしていますが
シフト演算で**溢れたビットは抹消**されます

シフト演算子は、算術演算子同様に2項演算子です
左シフト演算子は <<、右シフト演算子は >> と記述します

expr << count
expr >> count

expr には int、uint、long、ulong 型いずれかのシフトする式を指定します
count は int 型のシフトする回数を指定します

```
class Test {
 static void Main() {
  int x = 10;
  System.Console.WriteLine(((x << 2) + x) << 1);
  System.Console.WriteLine(x >> 1);
 }
}
```

2進数の各桁は2のウエイト(2のn乗)なので、シフトはある重要な意味を持ちます
すなわち左に1回シフトするごとに値は常に2倍となり
右に1回シフトするごとに値は半分になる(2で割る)ということになります

この性質を生かし、最初の WriteLine() は x を 10 倍して表示します
x を2回左シフトすることで 4 倍になり、これに x を加算することで5倍
さらに左に1回シフトすることで、x の値を10倍にしてメソッドに渡しています

下の WriteLine() メソッドは、x を1回右シフトすることで2で割っています
このような処理以外も、例えば上位ビットを削除したい場合や
24ビットで表される加法混色(RGB)の値で上位ビットの色の値だけを指定する時など
シフト演算はシステムプログラムやグラフィックの分野でも活躍します

算術シフト

ここで一つ問題になるのが、シフトすることでできた空きビットの値です
シフトで溢れた値が抹消されるのは問題ありませんが
その反対側でできた空きビットは一体どの値で初期化されるのでしょう

溢れたビットがシフト方向の反対側に出現するということはありません
ビットが回転する場合はシフトではなく「ローテート」と呼ばれる技術になります

通常は、シフトの場合反対側にできた空きビットは常に 0 で初期化します
このようなシフト演算を**論理シフト**と呼びます
しかし、論理シフトはビット列全体を対象にするため**最上位ビットが1**の場合
これは符号ビットなので、符号が反転してしまう可能性があります

そこで、右シフトをする時は**空ビットは符号ビットで初期化**されます
(もちろん、符号なしの型であれば符号ビットがないので論理シフトになる)
そうしなければ2の補数で表現されている負数をシフトで演算できないからですね
このようなシフト演算を**算術シフト**と呼びます

```
class Test {
 static void Main() {
  int x = -10;
  System.Console.WriteLine(x << 1);
  System.Console.WriteLine(x >> 1);
 }
}
```

コンパイルして実行すると、両方とも最上位ビットも数値も予想したとおりの結果になり
期待するシフト演算の意味を失うことはありません
