» 文法の曖昧さを理解して確実性と再利用性を高める

マイコンC言語 転ばぬ先のつえ

第16回　演算子⑦…論理演算子

鹿取 祐二

リスト1
ビットごとの論理演算子の優先順位は比較演算子や等値演算子よりも低い
誤った記述をしがちなので，常に（ ）を使うのがお勧め

```
unsigned int a;
    if( a & 0x0007 == 0x0001 )
```

（a）誤った使い方の例（式全体の評価結果はaに関係なく偽になる）

```
unsigned int a;
    if( ( a & 0x0007 ) == 0x0001 )
```

（b）（ ）を使った例

リスト2
ビットごとの論理演算子の中にも優先順位が存在する
AND，OR，XORの順番ではなく，AND，XOR，ORの順番であることに注意

```
unsigned int b;
    b = 1 & 1 | 0 ^ 1;
```

（a）誤った使い方の例（同一優先度で左結合性だと勘違いした場合）

```
unsigned int b;
    b = ( ( 1 & 1 ) | 0 ) ^ 1;
```

（b）（ ）を使った例

26 意外と優先順位が低い！ ビットごとの論理演算子

● 演算子の優先順位に注意すべし

&のAND，|のOR，^のXORのビットごとの論理演算子3つに関する注意事項は，シフト演算子と同様で，被演算数は正の値（ゼロを含む）の整数値に限ることです．文法では，符号付き整数型であっても正の値であれば移植性を損ないませんが，MISRA-Cでは符号なしの整数型のみに使用することを規定しています．

▶比較演算子や等値演算子よりも優先順位が低い

追加の注意事項としては，演算子の優先順位を注意することです．ビットごとの論理演算子は，思っているよりも優先順位が低いです．特に比較演算子や等値演算子よりも優先順位が低いのが特徴です．従って，**リスト1**（a）のような誤った記述をしがちです．

この場合，&よりも==の優先順位が高いので，先に`0x0007 == 0x0001`が評価されます．当然ですが，評価結果は偽なので0となり，次に`a & 0`が評価されます．0とANDは，他方の被演算数に関係なく0となるので，式全体の評価結果はaに関係なく偽になります．ビットごとの論理演算子と他の演算子を1つの式に記述するときは，**リスト1**（b）のように常に（ ）を使った方が良いでしょう．

● 演算子の中にも優先順位がある

▶優先順位はAND，XOR，ORの順番

3つのビットごとの論理演算子は，優先順位が異なることにも注意を払いましょう．乗除算や加減算の演算子とは異なり，ビットごとの論理演算子は同一優先度ではありません．しかも，AND，OR，XORの順番ではなく，AND，XOR，ORの順番です．これを誤ると**リスト2**（a）の式の結果は異なったものになります．

同一優先度で左結合性だと勘違いすると，1と1のANDで1，1と0のORで1，1と1のXORで0が変数bに代入されると解釈してしまいます．しかし，正しい評価順序は`b = ( 1 & 1 ) | ( 0 ^ 1 )`なので，1と1のANDで1，0と1のXORで1，1と1のORで1が変数bに代入されます．

AND，OR，XORの順序で演算したいのであれば，**リスト2**（b）のように（ ）を使って記述した方が分かりやすいでしょう．

27 副作用を伴う式は使用禁止！ 論理演算子

● 条件式の中身は演算されない場合がある

複数の条件を記述するための「なおかつ」を意味する&&と，「または」を意味する||は，どちらも論理演算子です．これらを使った結果は等値演算子と同じ1か0のどちらかです．型もint型と決まっているので，ほかの式に利用できます．ただし，次のような記述は難しいので，等値演算子などのように評価結果をほかの式に使うのは無理があるかもしれません．

```
a[ b == c && d < e ] = f;
```

そうなると，論理演算子は一般的な使い方しかできないでしょう．そのときの注意事項は，MISRA-Cの