≫ 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12回　演算子③…実は本当に必要なケースは多くない！キャスト演算子

鹿取 祐二

リスト1　キャスト演算子が必要になる場面…メモリ・マップドI/O方式マイコンの周辺機能を制御する場合
16進数で記述されたレジスタ番地をポインタ型に変換するのに使う

```
#define P1DDR  (*(volatile unsigned char *)0xFFFFC0)
#define P1DR   (*(volatile unsigned char *)0xFFFFC2)
```

本連載では，C言語の言語仕様（文法）の曖昧な部分と，それにより起こる問題を解説します．再利用性と効率が高く，安全かつ安心して使えるソフトウェアが開発できるようになることを目指します．

今回は，キャスト演算子の適切な使い方について解説します．〈編集部〉

■ キャスト演算子

● 諸悪の根源！乱用は避けよう

筆者は常日頃から，「算術変換規則を知らないCプログラマほどキャスト演算子を乱用する」と思っています．

確かにキャスト演算子が必要な場面はあります．しかし，本当に付けなければならない場面はそれほど多くありません．意味もなく付けられている場合が多いと感じます．

本連載で紹介してきた算術変換規則や定数値に対する接尾子を応用すれば，キャスト演算子はそれほど必要ではありません．逆に乱用すると，無駄な型変換が発生して，性能が劣化する原因にもなります．キャスト演算子は，算術変換規則を理解してから利用するようにしてください．

● 必要になるケース

キャスト演算子が必要になるのは，メモリ・マップドI/O方式を採用しているマイコンの周辺機能を制御する場合です．メモリ・マップドI/O方式では，周辺機能のレジスタは番地を指定して操作します．その場合，**リスト1**のように16進数などで記述されたレジスタの番地をポインタ型に変換しなければならないので，キャスト演算子が必要になります．

リスト2　キャスト演算子が不要な場面…関数の呼び出しを行う場合（システム・コール発行）
プロトタイプ宣言が存在する場合，実引数の型は自動的にプロトタイプ宣言で指定された型に変換されるので，キャスト演算子は不要

```
tk_wai_sem( (ID)1, (INT)1, (TMO)TMO_FEVR );
tk_set_flg( (ID)2, (UINT)0x03 );
```

（a）キャスト演算子が付いているが不要

```
ER tk_wai_sem(ID semid, INT cnt, TMO tmout);
ER tk_set_flg(ID flgid, UINT setptn);
```

（b）システム・コールのプロトタイプ宣言

● 不要なのに付いている例

関数の呼び出しを行うときの実引数に対してキャスト演算子を付けている場合があります．

例えば，ライブラリ関数やリアルタイムOSのシステム・コールを発行するときに，**リスト2**（a）のように行っているコードを見たことがあります．このキャスト演算子は不要です．理由は，プロトタイプ宣言が存在する場合，実引数の型は，**リスト2**（b）のプロトタイプ宣言で指定された型に自動的に変換されることになっているからです．

キャスト演算子を記述するのは無駄です．可読性は向上するかもしれませんが，それ以外に何も意味はありません．関数の実引数にキャスト演算子を記述するかどうかは，開発を始める前に方針を決めておくことをお勧めします．

■ 汎用ポインタ

● 便利だけど…やっぱり乱用は避けよう

キャスト演算子という意味では，汎用ポインタであるvoid *も乱用されることが多いと思います．

汎用ポインタは，どんなアドレス値でも代入が許される便利なポインタ型ですが，乱用は避けるべきです．必要もないのにvoid *を使うのは，「私はポインタを理解していません」と言っているのと同じです．

第3回　整数型③…整数の格上げと算術変換規則（2021年3月号）
第4回　浮動小数点数型①…内部表現と誤差（2021年5月号）
第5回　浮動小数点数型②…高速・高効率に演算する方法（2021年6月号）