≫ 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

新連載

第1回　整数型①…値の範囲と負の内部表現

鹿取 祐二

C言語は，誕生から50年近くたつ今でも，分野を問わず，さまざまな場面で使われるプログラミング言語です．言語仕様は標準規格化されていますが，曖昧な部分が多く存在します．

本連載では，C言語の言語仕様の曖昧な部分と，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．

第1回～第3回では，整数型の言語仕様に対する問題点を解説します．〈編集部〉

リスト1　C言語の曖昧な文法の例（char型を使った数値計算）
このプログラムは，実行環境によって結果が変わってしまう．原因は，char型の符号の有無が文法で決まっていないためだ

```
char a;

    a = 200;
    if( a == 200 )
        printf("char is unsigned char."); // RL78,RX
    else
        printf("char is signed char.");   // H8,SH
```

RL78とRXのコンパイラでは真になる

H8とSHのコンパイラでは偽になる

● 決して高くないC言語の移植性

C言語は誕生して約50年がたちます．当初はOS（UNIX）の記述言語として開発されました．現在ではその用途に限らず，組み込み系システムのソフトウェア開発では，ほとんどC言語が使われます．その理由はさまざまですが，当時としてはC言語以外に絶対番地の操作ができる言語が少なかったためと考えられています．そんなC言語ですが，昔から「移植性がない」と言われています．

誕生当初は「移植性が良い」との触れ込みでしたが，用途が拡大するにつれ，「本当に移植性が良いのか」とか「移植性はないだろう」という意見が聞こえるようになりました．ソフトウェア開発言語をアセンブリ言語からC言語に移行しようとしていた自動車業界も，移植性の低さから二の足を踏んでいたのは事実です．

そこで，MISRA（Motor Industry Software Reliability Association）がC言語のためのソフトウェア設計標準規格MISRA-Cを策定しました．MISRA-Cでルール化されたコーディング規約を守ることで，C言語で記述する組み込み系システムで安全性と可搬性（ポータビリティ，移植性），信頼性を確保できるようになりました．

● その理由は曖昧な言語仕様にあった

MISRA-Cは，C言語の言語仕様の曖昧な部分を使わないことで移植性の高いソフトウェア開発を目指しています．しかし，この方法だとプログラマの技術力向上につながらないと筆者は考えています．MISRA-Cを使っていても，なぜそのルールが存在するのか，守らないとどんな問題が発生するのかが分からず，中には問題の本質を知らないままプログラミングしている人もいるのではないでしょうか．数学に例えると，公式や定理は知っていても，その中身の本質を知らないまま使っているのと同じです．

C言語の曖昧な言語仕様の一例を**リスト1**に示します．char型の変数に200を代入していますが，このプログラムを実行すると，マイコンや処理系（コンパイラ）によって結果が真になったり偽になったりします．詳しくは後述しますが，この原因はchar型の符号の有無が言語仕様で決まっていないためです．**リスト1**のプログラムは，今の環境でたまたま真になっていても，移植したら偽になって正常に動作しなくなる可能性があります．

本連載では，このようにC言語の言語仕様の何が曖昧で，どのような問題が発生するのかを理解していきます．本連載を通して，文法の曖昧な部分を使っていても，**図1**のように移植性の高い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．

● マイコンC言語は言語仕様の理解力で差が出る

CPUの性能が高ければ，あまり言語仕様を意識せず，一番サイズの大きい型の変数を使えば済みます．プログラム・コードの貧弱さは高速なCPUがカバーし，データ・サイズの膨大さは大容量メモリが補います．