≫ 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7回　派生型②…構造体を使うときの作法とコード・サイズ削減テクニック

鹿取 祐二

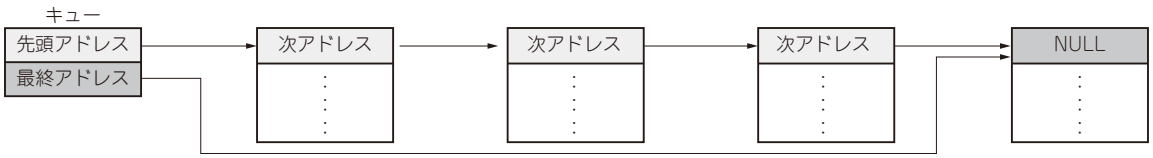

図1　構造体のリスト構造の例…自己参照構造体
構造体は非の打ち所がない文法を持つ．そのリスト構造は，OSのTCB（Task Control Block）などのキュー管理を始め，さまざまな用途に応用されている

C言語は，誕生から50年近く経つ今でも，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（文法）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．

今回は前回（本誌2021年7月号）に続いて派生型，いわゆる基本型から派生した配列，ポインタ，構造体，共用体，ビットフィールドについて紹介します．各派生型が移植性を重視した組み込み系システムで利用できるものなのかを理解しましょう．〈編集部〉

13 マイコンで構造体を使うときの作法とテクニック

● 構造体の文法は非の打ち所がない

構造体を使えば，新たなデータ型をいくらでも作り出すことができます．筆者はC言語の文法の中で構造体が一番優れていると思っています．

MISRA-Cにも構造体に対するルールはほとんど見つかりません．それほど非の打ち所がない文法であると言えるでしょう．特に自分自身へのポインタをメンバに持つ自己参照体型の構造体は**図1**に示すようなリスト構造，いわゆるOSにおけるTCB（Task Control Block）などのキュー管理にも使われ，さまざまな用途に応用されています．

例えば，キュー管理では，データ（テーブル）の登録や削除をFIFO（First In First Out）メモリやLIFO（Last In First Out）メモリで行うことが頻繁にあると思います．**リスト1**に示すのが，それらの代表的な関数郡です．文法的な問題はないので，特に詳しい説明は行いません．興味のある人だけ見ていただければよいかと思います．

● 組み込み系での作法…メンバはサイズの小さい順に宣言する

▶変数の配置場所の制約はマイコンごとに異なる

非の打ち所がない文法を持つ構造体ですが，組み込み系の場合は1つだけ注意しなければならないことがあります．それはアライメントです．

アライメントとは，変数が配置できる場所の制約のことで，マイコンの種類によって多少異なります．**表1**に示すのは，ルネサス エレクトロニクスのマイコンにおけるアライメントです．各変数は**表1**に示すアライメントでしか配置できません．

アライメントに違反したときの動作も，マイコンの種類によって異なります．H8やRL78の場合は，強制的に偶数番地に補正されます．SHの場合は，アドレス・エラーと言う例外が発生します．RXの場合はちょっと特殊であり，違反しても正しいデータは読み書きできますが，アライメントに違反しないときに比べてアクセス速度が劣化します．

▶宣言順が適当だとコード・サイズが増大する

このような制約から，処理系はアライメントに違反しないように変数を配置します．これが基本型の変数であれば，変数の配置順を宣言順とは異なるものとしたり，アライメントごとのセクションに分離したりするなどの対応が取れるのですが，構造体にはそれらの対策が施せません．理由は，構造体のメンバは宣言順

第1回　整数型①…値の範囲と負の内部表現（2021年1月号）
第2回　整数型②…表現できない値を代入したときの規則（2021年2月号）
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）