≫ 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22回　最適化①…実行速度と使用メモリ量はトレードオフ

鹿取 祐二

リスト1　100個の要素を持つint型の配列aを初期化するプログラム
繰り返し文を使うと，使用メモリ量は削減できるが実行時間が長くなる．一度に初期化する個数が多いほど実行時間は短縮できるが，使用メモリ量も増加する

```
int i;
    for( i=0 ; i<100 ; i++ )
        a[i] = 0;
```

（a）繰り返し文を使う

```
    a[0] = 0;
    a[1] = 0;
  // 途中を省略
    a[98] = 0;
    a[99] = 0;
```

（b）繰り返し文を使わない

```
int i;
    for( i=0 ; i<100 ; i+=2 )  {
        a[i] = 0;
        a[i+1] = 0;
    }
```

（c）1回の繰り返しで2個ずつ初期化

```
int i;
    for( i=0 ; i<100 ; i+=4 )  {
        a[i] = 0;
        a[i+1] = 0;
        a[i+2] = 0;
        a[i+3] = 0;
    }
```

（d）1回の繰り返しで4個ずつ初期化

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．

（編集部）

今回から，処理系が持つ最適化機構（以降，最適化と記す）について紹介します．最適化はプログラムのブラッシュアップを行う反面，バグの原因にもなり得ます．また，バグを恐れるあまり，最適化を抑止するvolatile型修飾子をやみくもに利用するのも何かと問題があります．そうならないためにも，ここでは最適化に関する基本的な内容を解説します．

まず最適化で覚えてほしいのは，実行速度向上の最適化と使用メモリ量削減の最適化は反比例の関係にあることです．もちろん，両方の効率が向上するのであれば，処理系は積極的に最適化を行います．しかし，そうでない場合はどちらかに的を絞って最適化を行うことになります．

● 繰り返し文の有無で考えてみる

例えば，100個の要素を持つint型の配列aを初期化する場合，**リスト1**のどちらのプログラムが優れているでしょうか．

リスト1（a）はfor文を使って初期化しています．一方，**リスト1**（b）は繰り返し文を使わず，直接100個の要素を初期化しています．使用メモリ量，つまりコード効率が優れていることから，ほとんどの人は**リスト1**（a）の方が優れていると思うでしょう．しかし，実行速度は**リスト1**（b）の方が優れています．

リスト1（a）は，ループ・カウンタ用変数iの操作が必要で，実行時間のかかる分岐命令も必要です．それらが必要ない分，**リスト1**（b）の方が実行速度が速いのです．

もっとも**リスト1**（b）はかなり極端な例です．もし製品への適用を検討するならば，**リスト1**（c）や**リスト1**（d）のような記述を考慮すべきです．

リスト1（a），（c），（d）の実行速度と使用メモリ量の比較を**表1**に示します．どれも同じような結果であり，実行時間と使用メモリ量の性能は反比例の関係であることが理解できます．

第10回　演算子①…「関数呼び出し」に使う小括弧（2021年12月号）
第11回　演算子②…[ ] と * の意味，++a とa++ の違い（2022年1月号）
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（2022年2月号）