誰もがハマってしまう？ C言語の思わぬ落とし穴

舘 伸幸

C言語は自由度が高い反面，良く考えてプログラムを記述しないと思った通りに動作しないことがある．また，メモリ配置のようにC言語の言語仕様以外の部分の知識がないと，正常に動作しないこともある．ここでは，C言語を使ううえで思わず「ハマって」しまう落とし穴について解説する．
（編集部）

1．ポインタと配列の問題

第1章で説明したとおり，C言語はほかの高級言語と異なり，低水準性を持っています．自由度の高いプログラミングができることと引き替えに，常に次の二つのことを意識してプログラミングしなければ，思わぬ落とし穴にはまる可能性があります．

一つ目は，メモリという物理リソースを常に意識する必要があるという点です．二つ目は，C言語の記述が，実際にはどのような実行プログラムに翻訳されるのか知っている必要があることです．では，順番に見てみましょう．

● 見た目は似ていても全く異なる宣言文

初めに，ポインタと配列の問題について解説します．一般に，C言語ではポインタが難しいといわれますが，ポインタ単体よりも，配列との関係で混乱してしまうことが主な原因です．そしてその混乱の原因は，メモリのイメージが頭の中にできていないことによります．

リスト1に示す二つの宣言式を見てみましょう．

この二つの宣言文の見た目はよく似ていますが，文法上も実体上も全く異なります．①の"CQpublishing"は，文字列リテラルと呼ばれ，一種の定数です．メモリ上のどこかに実体が存在して，ポインタはその最初の文字である'C'を指すように初期化されます．

リスト1　二つの宣言式

```
char *str1  = "CQpublishing";  ←①
char str2[] = "CQpublishing";  ←②
```

一方②の"CQpublishing"は，配列str2[]の初期化データです．これは書き換えれば，

```
char  str2[]  =  {  'C','Q','p','u','b',
'l','i','s','h','i','n','g','¥0' };
```

と同じ意味です．

メモリのどこかにstr2という名前の配列の領域が確保され，その中身が"CQpublishing"で初期化されるのです．つまり，①では"CQpublishing"の文字列とは別にstr1というポインタ（変数）があるのに対して，②は，文字列そのものがstr2という配列（の中身）なのです（**図1**）．

さらに，①での文字列がメモリのどのような場所に確保されるかはコンパイラの処理次第（処理系定義）です．メモリは，読み出し専用のROMと読み書きできるRAMの2種類の領域に分かれています．文字列リテラルがどちらの領域に確保されるかは，コンパイラ次第なのです．そのため，①の宣言の後，ポインタstr1を使って内容を書き換

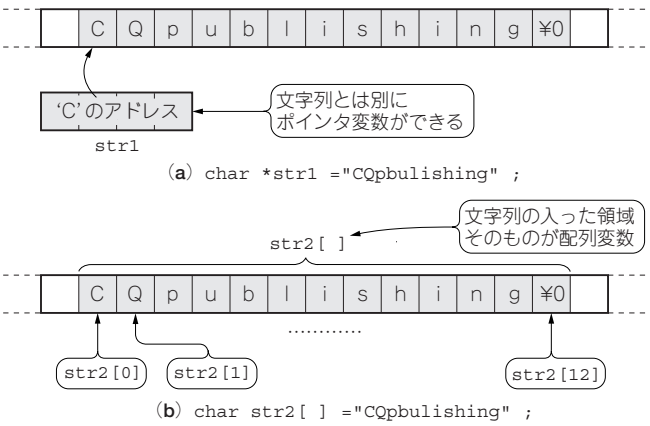

図1　ポインタと配列ではメモリ・イメージが異なる