マイコンを正しく操縦するための作法

基礎から学ぶC言語講座

岡田 好一
Yoshikazu Okada

第2回 算術演算と制御構造をマスタしよう

今回は，C言語でプログラムを作成するうえで欠かせない算術演算と制御文について，R8C/Tinyの特徴を交えながら解説します．

算術演算と型

C言語の算術演算で扱う「型」は「整数」と「浮動小数点数」に大別されます．

● 自然な語長の符号付き整数int型

R8C/Tinyは16ビット機ですから，0～65535または－32768～32767(2の補数表現)の範囲の整数がもっとも扱いやすく，機械語，つまり論理回路もそのように設計されています．

C言語のint型(intはinteger ＝ 整数の略)はターゲット機の自然な語長の符号付き整数ですから，R8C/Tinyでは－32768～32767がint型です．

型(type)はメモリ上のビット・パターンを計算機言語でどのように扱うかの規定です．コンパイラが機械語を選択する際に必要な情報です．後述する「変数」では明示する必要があります．定数にも型がありますが，明示する場合は限られています．

0～65535として使う場合は符号なしなので，「unsigned int」と，unsignedを前に付けます．すべてのビットが対等なので，後述するビット演算では符号なし整数が推奨されます．C言語の符号なし整数は桁あふれしません．16ビットの場合，65535の次は0で，0の前は65535です(**図2－1**)．

値	ビット・パターン
65535	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
65534	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
65533	1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1
65532	1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
65531	1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
65530	1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0
65529	1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1
⋮	⋮
6	0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
5	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
4	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
3	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
2	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

図2－1　符号なし16ビット整数のビット・パターン

Keyword　演算対象，数学的帰納法

● 演算対象(operand)

機械語命令内のレジスタやメモリを指す部分．この英語も訳しにくい．元の意味は演算子(operator)の対象．operandにoperation(演算)するとresult(結果)が得られる．「演算対象」では命令内部のニュアンスが出ないので，無理に訳さずに(命令の)オペランドと記されることも多い．

● 数学的帰納法(mathematical induction)

帰納法は個々の事例から一般的な結論を導き出すこと．数学的帰納法は最初の真の命題を提示した後，前の命題が成り立てば次の命題が成り立つことを証明し，一般に成り立つことを表明する論法．計算機言語の再帰と(向きは逆だが)話の進めかたが似ているのは気のせいではない．再帰の利用法を深く知りたい方はLISPの一方言Schemeや Prolog言語を調べてみるとよいだろう．

本文には記さなかったが，fact関数は階乗(factorial)を求める関数である．再帰の練習としてよく取り上げられる．こんな簡単な再帰関数でも，効率化のテクニックがある．

2の補数とは，負数の場合に全ビットの一つ上位に1を立てた数(16ビットなら2^{16} = 65536)から絶対値を引いたものです．そのように定義すると，最上位ビットが符号ビットになる，加算回路が符号なしと共通になる，という性質から「2の補数」は広く採用されています．符号付きを明示する場合は「signed int」とします(**図2-2**)．

数学的な整数(無限！)に比べて桁数が極端に少なく思えますが，もっぱら計数や配列の添字や識別番号として使うので，多くの場合に十分です．

● 変数には局所変数と大域変数がある

電卓で言えばメモリに相当する記憶は，C言語などの計算機言語では「変数」と呼ばれます．

変数を定義するには，すなわち主記憶上に領域を確保するには，プログラム中で使用前に，

値	ビット・パターン
32767	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
32766	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
32765	0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1
2	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
−1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
−2	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
−3	1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1
−32766	1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
−32767	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
−32768	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

図2-2　符号付き16ビット整数のビット・パターン(2の補数表現)

```
int i;
```

などと宣言を書きます．

C言語の変数名は英字で始まり，英数字の続く文字列です．大文字と小文字は区別されます．C言語では下線(_)も英字扱いです．長さは31文字以内です．

この宣言をmain関数の中に書くのと，外に書くのとでは大違いです．中に書くとmain関数内でしか使えません．これを局所変数と呼びます．外に書くとそれ以下のプログラム中どこでも(ほかの関数内でも)使えるので，大域変数と呼びます．

C言語に慣れてくれば，局所変数を多用することになります．大域変数はデータベース的なデータとなるでしょう．

一般に，変数の参照が可能なソース・プログラム上の範囲を「スコープ」と言い，日本語では単に「有効範囲」と表します．変数が参照できる時間の範囲を「記憶域期間」と言います(マニュアルでは「生存区間」)．普通に宣言すると，大域変数はプログラムの実行中はずっと生きていて，局所変数は関数の開始から終了までです．ただし，static指定子が付いた局所変数は，プログラム実行中は生存しています．C言語の変数は，型，スコープ，記憶域期間の3属性で使いかたが決まります．

変数名，型，スコープ，記憶域期間を知っているのはプログラマとコンパイラだけです．マイコン上に転送された機械語プログラムに埋め込まれてしまうので，それらの情報はデータとしては残りません．

● 算術演算に使われる演算子と優先順位

演算結果は通常，記憶させるか出力するかです．変数への「代入」とは，右辺の演算結果の左辺の変数への上書きです．変数の前値は残りません．連載の後半で述べるように，R8Cでは出力も特別なレジスタへの書き込みなので，C言語では代入になります．C言語の代入は「=」演算子を使います．

リスト2-1を見れば何となくわかると思いますが，

Keyword　サブルーチン，スタック

● サブルーチン(subroutine)

何度も使われる動作をまとめて，ほかのルーチン(プログラムの一部)から呼び出せるようにした計算機言語のしかけ．C言語では「関数」がサブルーチンの役目を果たす．対立語はメイン・ルーチン(プログラムの主要部分)で，C言語のmain関数は，ここがメイン・ルーチンですよ，ということ．

● スタック(stack)

積み重ねの意味で，上からしか出し入れできないので，LIFO(Last In First Out)つまり後入れ先出しのデータ構造を指す．一般のデータ構造としてのスタックには数式の解析など，いろいろ応用がある．

CPUのスタックは，サブルーチンや割り込みルーチン呼び出し時に使われる，ハードウェアが直接に支援するスタック構造を指し，そのスタック構造が使う主記憶領域も指す．データを入れる操作をpush，出す操作をpopと呼び，R8Cにも同名の命令がある．関数の戻り番地や引数や局所変数はここにはかなく生きていて，つまりはC言語などの関数呼び出し構造は，1本のスタック構造で管理できる，ということ．