特設

第7章 C言語で一番難しい文法だが…基礎はそれほど難しくない

配列を指すときに威力を発揮する「ポインタ」

鹿取 祐二

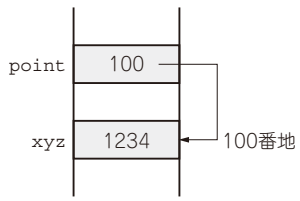

図1　ポインタにアドレスを格納した後の各変数の関係
ポインタ変数pointは変数xyzのアドレスを指していることを意味する

世間一般では，C言語で一番難しい文法はポインタだと言われています．これは紛れもない事実だと思います．とは言えポインタは，使い方によっては限りなく難しくなりますが，基礎はそれほど難しくありません．本章では，ポインタの基礎を紹介します．

1 ポインタの基本

● 宣言の記述方法

ポインタは，アドレスを格納する変数です．メモリ内に配置された変数や関数は，全てアドレスを持っています．そのアドレスを格納するのがポインタです．ポインタの宣言は次のように記述します．

```
型　＊名前;
型　＊名前, ＊名前, ＊名前;
```

ポインタは，型を記述したら，「名前」の前に＊を付けます．もし複数のポインタを一度に宣言するのであれば，カンマで区切って，名前の前に＊を付けます．もし，＊を付け忘れると，単なる基本型の変数宣言になってしまいます．例えば次のように記述すれば，ポインタ変数pointを宣言したことになります．

```
int  *point;
```

ポインタの宣言で重要なことは，＊の前に指定してある型です．と言うのも，ポインタは全ての変数や関数のアドレスを格納できるわけではありません．宣言時に指定した型以外の変数や関数のアドレスは格納できません．例えば前述のポインタ変数pointはint型なので，int型変数のアドレスしか格納できません．

● アドレスを代入する方法

ポインタにアドレスを格納するとは，言い換えればポインタ変数にアドレスを代入することです．変数のアドレスは&のアドレス参照演算子を使うことで取り出せます．例えば次のように記述すれば，int型の変数xyzのアドレスを取り出して，ポインタ変数pointに格納できます．

```
int  xyz = 1234;
int  *point;

     point = &xyz;
    // 変数xyzのアドレスを変数pointに代入
```

この結果，各変数の関係は**図1**の通りになります．図中に記載されている変数の番地は単なる例なので，実際の番地との関係などはあまり気にしないでください．変数の番地は対象のマイコンや開発環境によって変化します．それよりも重要なことは，ポインタにアドレスを格納したことの意味です．**図1**に記載した通り，ポインタが目的の変数を指したことを意味します．

● アドレスを格納した変数を間接的に操作できる

ポインタがアドレスを格納すると何がうれしいのでしょうか．答えは，アドレスを格納した変数を間接的に操作できることです(**図2**)．

例えば，**図3**であればポインタ変数pointを使って，変数xyzの内容が操作できます．具体的には次のように記述すれば，ポインタ変数point経由で変数xyzに5を代入できます．

```
  *point = 5; // 変数xyzに5を代入
```

見た目で分かるように，前述の式には変数xyzが登場しません．それでも操作されているのは変数xyzです．つまり，ポインタに格納したアドレスを変化させることで，同じ記述でありながら異なる変数を操作できることがうれしい点です．

ここで使われている＊演算子は，掛け算を意味する乗算演算子ではありません．乗算演算子は二項演算子(演算子の両側が被演算数)ですが，ポインタの指す内容を操作する間接演算子は左側が項でない単項演算