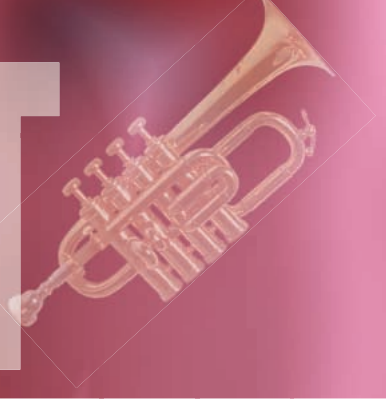
Appendix 1

# C言語の文法をこっそり復習しよう

増田 晃章，吉田 悦康，奈良 久美子



ここで，「C言語は習ったはずなのだけれど，もう忘れてしまったかも…」，という方のために，C言語の文法をまとめてみます．みんなには内証で，読み飛ばすふりをしてこっそり読んでください．**リスト1**と**リスト2**を例に説明していきます．

## ● 基本の基本 —— 型と変数について

定数に対して，データを格納するための入れ物（メモリ上の領域）のことを変数といいます．プログラム中で変数を使用する場合は，変数の宣言（メモリ上の領域を確保すること）を行う必要があります．変数宣言は，プログラムの先頭で，処理を記述する前にまとめて行います．

**（例1）変数の宣言**

```
型指定子　変数名；
```

変数宣言時は，格納するデータの種類に対応した型指定子を指定する必要があります．型指定子には**表1**のような種類があり，扱えるデータの範囲とメモリ上に確保される変数領域の大きさが異なります．

## ● 枝分かれと繰り返し —— if，switch，while，for

**（例2）if文**

```
if(条件式){
    処理A ;
}
else{
    処理B ;
}
```

条件により2分岐する場合は，if 文を使用します．条件式が真ならば処理Aを実行し，偽ならばelse以降の処理Bを実行します．

**（例3）switch文**

```
switch(式){
    case a:
        処理A ;
        break ;
    case b:
        処理B ;
        break ;
    default:
        処理N ;
        break ;
}
```

条件により複数に分岐する場合は，switch文を使用します．

**表1　型指定子の種類**

格納するデータの種類に対応した型指定子を指定する必要がある．

| 型指定子 | | | バイト数 |
|---|---|---|---|
| 数値 | 整数 | short | 2バイト |
| | | int | 2バイト<br>4バイト |
| | | long | 4バイト |
| | 実数 | float | 4バイト |
| | | double | 8バイト |
| 文字 | char | | 1バイト |

CPUのビット・サイズにより，下記のようにサイズが異なる
- 16ビットCPUのときは2バイト
- 32ビットCPUのときは4バイト

式がaならば処理Aを実行し，bならば処理Bを実行します．そして，どれにもあてはまらなければ，処理Nを実行します．コロン（:），セミコロン（;）を間違えないよう，break;を忘れないよう注意しましょう．

**（例4）while文**

```
while（条件式）{
    処理A ;
}
```

処理を繰り返す場合は，while文を使用します．条件式が真ならば，処理Aを繰り返し実行します．ちなみに，0を偽，1を真と判定するので，while(1)は無限ループとなります．

**（例5）for文**

```
for（初期設定 ; 条件式 ; 後処理）{
    処理A ;
}
```

初期設定や後処理を指定して処理を繰り返す場合は，for文を使用します．条件式が真ならば，処理Aと後処理を繰り返し実行します．

## ● 関数 —— 引き数と戻り値

**（例6）関数の定義と呼び出し**

```
b = 関数名( a ) ;
```

関数とは，必要なデータ（a：引き数と呼ぶ）を受け取り，何らかの処理を行って，結果（b：戻り値と呼ぶ）を返すものです．プログラムの中で実体（どのような処理を行うかという具体的な内容）を定義しておき，別の場所からその関数を呼び出す形で処理を実行します．

## ● 配列 —— 変数と配列の違い

**（例7）配列の宣言**

```
型指定子　配列名［ 要素数 ］;
```

配列は変数の一種です．一つの変数で扱えるのは一つのデー



KEYWORD —— C言語，if文，switch文，while文，for文，printf，scanf，strcpy，strcat

リスト1　例1～例6

```
#include "common.h"        /* 共通ヘッダ・ファイル */
#include "fr.h"            /* I/O領域参照宣言 */

void voShiftLed( void )                          ← 関数の実体定義（例6）
{
    uchar ucData;
    uchar ucLed_cnt;                             ← 変数の宣言（例1）
    uchar ucLed;
    int i;

    ucLed_cnt = 8;

    while(1){                                    ← while文（例4）
                                                 ← if文（例2）
        /* LEDをシフト */
        if( ucLed_cnt < 8 ){
            ucData <<= 1;           /* 1ビット左へシフト */
            IO_PDR2.byte = ucData;  /* LEDを点灯 */
            ucLed_cnt++;
        }
        else{
            ucData = 0x01;
            IO_PDR2.byte = ucData;  /* LED2を点灯      */
            ucLed_cnt = 1;
        }

        /* LEDの点灯間隔を変更 */
        switch( ucLed_cnt ){        /* カウンタを判定 */
            case 1,2,3:
                Wait(WAIT_COUNT1);
                break;
            case 4,5:
                Wait(WAIT_COUNT2);
                break;
            default:
                Wait(WAIT_COUNT3);
                break;
        }
    }

    for( i=0; i<10; i++ ){                       ← switch文（例3）
        /* LEDを点灯 */                          ← for文（例5）
        ucLed = ucSet_led( i );

        /* 7SEGの点灯を0～9へ変動させる */
        voSet_7seg( i, ucLed );
    }
    return;
}                                                ← 関数の呼び出し（例6）
```

リスト2　例7～例12

```
#include <stdio.h>                               ← include文（例12）

void    voOutputFilename( void )
{
    FILE    *ifp;
    FILE    *ofp;
    char    cFname1[64];                         ← 配列の宣言（例7）
    char    cFname2[64];
    int     c;

    printf( "複写元のファイル名=>" );              ← printf関数の呼び出し（例8）
    scanf( "%s", cFname1 );                      ← scanf関数の呼び出し（例9）

    ifp = fopen( cFname1, "r" );                 ← fopen関数の呼び出し（例13）

    printf( "複写先のファイル名=>" );
    scanf( "%s", cFname2 );

    ofp = fopen( cFname2, "w" );                 ← getc関数の呼び出し（例15）

    while( getc( ifp ) != EOF ){
        c = getc( ifp );                         ← putc関数の呼び出し（例16）
        putc( c, ofp );

        strcpy( text1, text2 );                  ← strcpy関数の呼び出し（例10）
        printf( "¥n%s¥n", text1 );

        strcat( text1, text2 );                  ← strcat関数の呼び出し（例11）
        printf( "¥n%s¥n", text1 );
    }

    fclose( ifp );                               ← fclose関数の呼び出し（例14）
    fclose( ofp );

    return;
}
```

タのみです．これに対して配列では，同じ型のデータが複数ある場合，それらをひとまとめにして扱うことができます．特に，同じ型の関連するデータを同じ名前で利用できる点が便利です．

## ● 文字列を扱う――文字の入出力，コピー，連結

**(例8) 文字列の出力**

```
printf （ 文字列 ） ;
```

画面に文字列を表示するには，主にprintf関数を使用します[注1]．画面に表示するデータを引き数として文字列で指定します．ここでいう文字列はC言語で扱うデータの一種に当たり，ダブルクォーテーション（"）で囲みます．

**(例9) 文字列の入力**

```
scanf （ 書式 ， 変数のアドレス ） ;
```

キーボードからデータを受け取るには，主にscanf関数を使用します．scanf関数は，第1引き数（1番目の引き数）に受け取るデータの書式を指定し，第2引き数（2番目の引き数）にデータを格納する変数のアドレスとして「&変数名」を指定します．scanf関数は，キーボードから受け取ったデータを指定した書式に従って変換し，変数に格納します．

**(例10) 文字列のコピー**

```
strcpy （ コピー先のアドレス ， コピー元のアドレス ） ;
```

文字列を複写（代入）する場合は，主にstrcpy関数を使用します．strcpy関数は，文字列をコピー元のアドレス（第2引き数）からコピー先のアドレス（第1引き数）に複写します．

**(例11) 文字列の連結**

```
strcat （ 連結先のアドレス ， 連結元のアドレス ） ;
```

文字列を連結するには，主にstrcat関数を使用します．strcat関数は，連結先アドレス（第1引き数）の文字列の後に，連結元アドレス（第2引き数）の文字列を連結します．

## ● コンパイルの前にヘッダ・ファイル ―― include

**(例12) include文**

```
#include "ヘッダ・ファイル名"
#include <ヘッダ・ファイル名>
```

注1：printf，scanf，strcpy，strcatといった関数は，C言語の入門書には必ず記載されている代表的なものである．しかし，特に異常系処理などにおいて幾つかの問題を指摘されていることも事実であり，製品開発においては注意が必要．