# Lesson 2
# 表示に凝ろう

本 Lesson では《暗算能力チェック》《テロップ》など、時間をあやつったり、文字を消したり動かしたりするプログラムの作成を通じて、時間や表示に関する各種のテクニックを身につけます。

## この Lesson で学ぶおもなこと

- 拡張表記
- 警報
- 改行と復帰
- 後退
- タブ
- 引用符
- 表示済み文字の消去と書きかえ
- 一定時間の処理停止
- 処理時間の計測
- キャスト
- 文字列
- ナル文字

- clock_t 型
- clock 関数
- printf 関数
- putchar 関数
- strlen 関数

# 2-1 拡張表記を使いこなす

拡張表記を活用すれば、画面上の文字を消したり動かしたりと、さまざまな表示効果を生み出せます。まずは、拡張表記をひと通りマスターしましょう。



## 拡張表記

**拡張表記**（*escape sequence*）は、逆斜線記号 \ を先頭にした文字の並びによって、単一の文字を表す表記法です。その一覧を **Table 2-1** に示します。

■**Table 2-1**　拡張表記（escape sequence）

| | | |
|---|---|---|
| ■ **単純拡張表記**（*simple escape sequence*） | | |
| `\a` | **警報**（*alert*） | 聴覚的または視覚的な警報を発する。 |
| `\b` | **後退**（*backspace*） | 表示位置を直前の位置へ移動する。 |
| `\f` | **書式送り**（*form feed*） | 改ページして、次のページの先頭へ移動する。 |
| `\n` | **改行**（*new line*） | 改行して、次の行の先頭へ移動する。 |
| `\r` | **復帰**（*carriage return*） | 現在の行の先頭位置へ移動する。 |
| `\t` | **水平タブ**（*horizontal tab*） | 次の水平タブ位置へ移動する。 |
| `\v` | **垂直タブ**（*vertical tab*） | 次の垂直タブ位置へ移動する。 |
| `\\` | **文字** \ | |
| `\?` | **文字** ? | |
| `\'` | **文字** ' | |
| `\"` | **文字** " | |
| ■ **8進拡張表記**（*octal escape sequence*） | | |
| `\ooo` | *ooo* は 1 ～ 3 桁の8進数 | 8進数で *ooo* の値をもつ文字。 |
| ■ **16進拡張表記**（*hexadecimal escape sequence*） | | |
| `\xhh` | *hh* は任意の桁数の16進数 | 16進数で *hh* の値をもつ文字。 |

これらを学習して使いこなせるようにしましょう。

▶　拡張表記 `\n` や `\x1B` の見た目は 2 文字以上ですが、それによって表されるのは、あくまでも 1 文字です。

## \a … 警報

警報 `\a` を出力すると、〔**聴覚的または視覚的な警報**〕が発せられます。ほとんどの環境ではビープ音が鳴ります（音を出さずに画面を点滅させる環境もあります）。

なお、警報を出力しても**現表示位置**（コンソール画面における**カーソル**の位置）が変更されることはありません。

▶　p.2 でも述べたように、本書では警報の出力結果を♪と表します。

## \n … 改行

改行 \n を出力すると、現表示位置が〔**次の行の先頭**〕に移動します。
警報と改行を出力するプログラム例を **List 2-1** に示します。

**List 2-1**

```
/*
    警報\aと改行\nを出力する例
*/

#include <stdio.h>

int main(void)
{
    printf("こんにちは。\n初めまして。\n");
    printf("\a警告します。\n\n");
    printf("\a\a今度は２回警告します。\n");

    return (0);
}
```

途中で改行

１行空ける

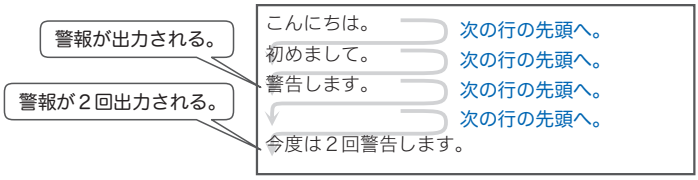
警報を発する

**実行結果**

```
こんにちは。
初めまして。
♪警告します。

♪♪今度は２回警告します。
```

『こんにちは。』に続いて改行文字 \n を出力するため、**Fig.2-1** に示すように『初めまして。』は次の行に表示されます。




● **Fig.2-1**　改行文字の働き

２番目に表示する文字列の末尾は、改行が二つ続いた "\n\n" です。そのため、最後の出力は、１行空いたところに表示されます。

▶　このテクニックは、Lesson 1 の数当てゲームでも利用していました。

## \f … 書式送り

書式送りを出力すると、現表示位置が〔**次の論理ページの先頭位置**〕に移動します。通常の環境では、書式送りをコンソール画面へ出力しても何も起こりません。

**プリンタへの出力において、改ページを行う際に利用します。**

## \b … 後退

後退 \b を出力すると、現表示位置が〔**その行内での直前の位置**〕に移動します。

▶ 行の先頭に現表示位置があるときに後退を出力した結果は規定されていません。多くの環境では、前の行（上の行）にはカーソルを戻せないからです。



後退によって**動きのある表示効果**を生み出す例が **List 2-2** です。"ABCDEFG" と表示して、それから 1 秒ごとに末尾側から 1 文字ずつ消すプログラムです。

**List 2-2**

```
/*
    後退\bの利用例 …  1秒ごとに1文字消去
*/

#include <time.h>
#include <stdio.h>

/*--- xミリ秒経過するのを待つ ---*/                    xミリ秒だけ時間をつぶす関数
int sleep(unsigned long x)
{
    clock_t c, s = clock();

    do {
        if ((c = clock()) == (clock_t)-1)       /* エラー */
            return (0);
    } while (1000UL * (c - s) / CLOCKS_PER_SEC < x);
    return (1);
}

int main(void)
{
    int i;

    printf("ABCDEFG");

    for (i = 0; i < 7; i++) {
        sleep(1000);                    /* 後ろから1文字ずつ */
        printf("\b \b");                /* 1秒ごとに消す     */
        fflush(stdout);                 /* バッファを掃き出す */
    }

    return (0);
}
```

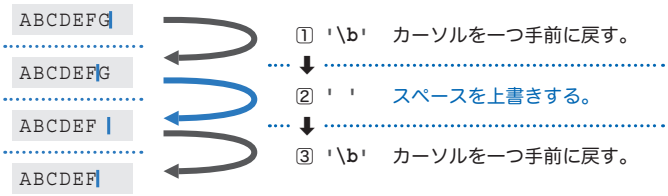
出力を確実に行う


関数 *sleep* の働きは、*x* ミリ秒だけ**時間をつぶす**ことです。ここでは *sleep*(1000) と呼び出されていますので、1 秒たってから呼出し元に戻ります。

▶ この関数の詳細は p.40 で詳しく解説します。

**main** 関数では、まず "ABCDEFG" を表示します。それから 1 秒ごとに "\b \b" を出力して、後ろ側から一つずつ文字を消します。

文字を消す仕組みを **Fig.2-2** を見ながら理解しましょう。




● **Fig.2-2**　後退による文字の消去

文字列 "ABCDEFG" を表示した時点で、カーソルは文字 'G' の直後に位置しています。この状態から、以下の手順で文字 'G' を消去します。

① 後退 '\b' を出力：カーソルが一つ戻って 'F' の直後へ移動する。
② 空白 ' ' を出力：文字 'G' が消える。
③ 後退 '\b' を出力：カーソルが再び一つ戻って 'F' の直後へ移動する。

ただし、この出力を命じた結果が即座にコンソール画面上に反映されるとは限りません。そこで、***fflush*** 関数を呼び出して出力を確実にします。

▶　プログラムで出力を指示するたびに画面やファイルなどへ書き込んでいるのでは、高速な処理は行えません。そのため、多くの処理系・環境では、出力すべき文字をバッファにためておき、『改行文字の出力が指示された。』とか『バッファが満杯になった。』といったことを契機に実際の出力が行われるようになっています。

出力した "\b \b" がバッファに蓄えられたままだと文字は消えません。確実に文字を消すためには、***fflush*** 関数によって強制的にバッファの内容をフラッシュする（掃き出させる）必要があります（***fflush*** 関数の呼出しが不要な環境もあります。しかし、この呼出しを省略すると、文字が即座に消えるかどうかが、環境に依存することになります）。

なお、ストリームやバッファ、さらに ***fflush*** 関数などについては、Lesson 9 で詳しく学習します。

末尾の文字 'G' の消去が完了したら、同じ手順によって、文字 'F', 'E', … と後ろから順に 1 文字ずつ消していきます。全部の文字を消すとプログラムは終了です。

*

関数 *sleep* に渡している値 1000 を変更すると、**文字を消すスピードが変わります**。プログラムを書きかえて試してみましょう。