第7章

関数呼び出し時の引き数/戻り値の受け渡しや再起呼び出しまで

関数呼び出しとスタックの関係を知ろう

C言語のプログラムは「関数」と呼ばれる処理の単位にまとめられている．この章では，関数の基本的な使い方について説明し，関数呼び出しに伴うプロセッサの内部動作を解説する．（筆者）

三好 健文

1．関数の超基本的な記述方法

関数はC言語における処理の単位です．**図1**に示すように，関数は「入力」を与えると「出力」を返します．C言語では，入力する値を「引き数」，出力する値を「戻り値」と呼びます．

関数は，次のように記述されます．

```
戻り値の型 関数名（仮引き数）
{
  処理の本体
    ...
  return 戻り値;
}
```

●戻り値の型

intやcharなどのプリミティブ型のほか，構造体やポインタ型を用いることができます．C言語では，関数の戻り値は一つだけです

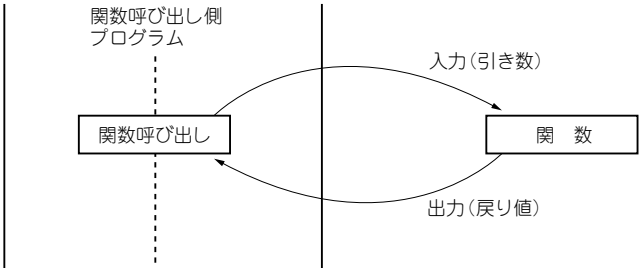

図1　関数呼び出しの関係

●関数名

変数名と同じようにアルファベット，数字，「_」を使用し名前を付けます．ただし，最初の1文字を数字にすることはできません．

●仮引き数

必要な変数の型と名前を定義します．複数の引き数を受け取る場合は，仮引き数を「,」で区切って列挙します．

● int型の関数の書き方

int型の変数xとyを受け取りその和を返す関数は，下記のように記述されます．

```
int func(int x, int y)
{
  int z;
  z = x + y;
  return z;
}
```

関数funcが呼ばれたとき，int型の二つの値が関数に与えられます．関数中の処理では，与えられた値を仮引き数で指定した変数名（xとy）に代入し処理します[注1]．処理の最後のreturn文によって，戻り値に指定したint型の値zを呼び出し，元に返します．

● void型の関数の書き方

戻り値を持たない関数は，void型の関数で定義します[注2]．例えば第2章のリスト2で，LEDに値を代入するた

注1：仮引き数と関数の中で宣言する変数の名前は重複していてはいけない．
注2：voidは，「何も返さない」という意味で付けられた．

KEYWORD——引き数，戻り値，仮引き数，実引き数，フレーム，スタック

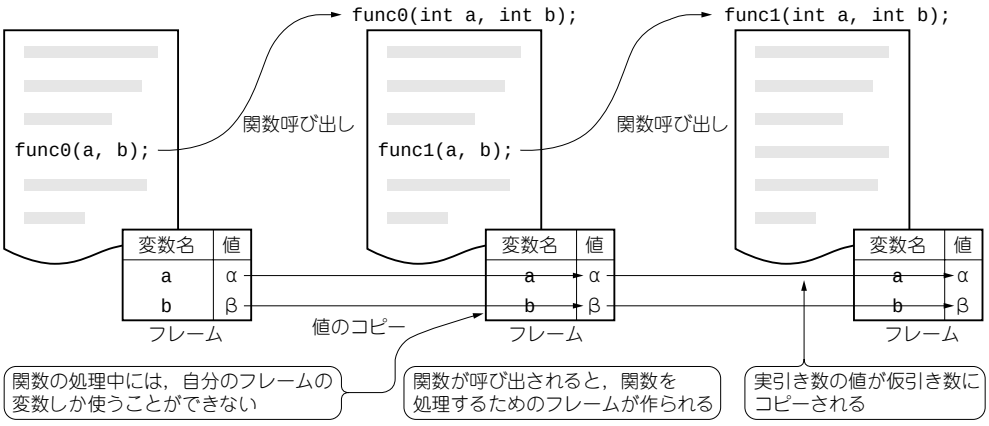

図2
関数呼び出しに伴うフレーム生成

めの関数 setLED に void 型を利用しています．この関数では，仮引き数 value に対して与えられた値を LED に代入するだけのプログラムで，明確な戻り値を持っていません．

```
void setLED(char value) {
  char *LED    = (char *)0x40000004;
  *LED = value;
}
```

戻り値のない関数でも，引き数なしの return 文を使えば明示的に「関数から戻ること」を記述できます．return 文があるとその後のプログラムは実行されずに関数の処理がそこで終了して，呼び出し元に戻ります．

2. 関数呼び出し側の記述方法

定義した関数を呼び出すためには関数を呼び出す側で下記のように記述します．

```
関数名(変数に与える値);
```

関数を呼び出す際に変数に与える値を「実引き数」と呼びます．実引き数は，呼び出す関数の仮引き数で定義されている個数分必要です．複数の値を並べる場合には，「,」で列挙します．例えば，先の関数 func を呼び出すには下記のようになります．

```
func(10, 20)
```

● 関数のプロトタイプ宣言

関数を呼び出す側は，呼び出される関数がどのような戻り値の型，仮引き数で定義されているのかを知ることはできません．C 言語では「プロトタイプ宣言」により関数がどのように定義されているのかを記述することができます．プロトタイプ宣言は，変数名や戻り値の型，仮引き数を次のように記述します．

```
戻り値の型 関数名(仮引き数);
```

どこかで見た形ですね．これは先の関数の定義から「{処理の本体}」を除き，また文末に「;」を付けた文と同じです．一般に，複数の C 言語ソース・コードで共有される関数のプロトタイプ宣言は，ヘッダ・ファイルにまとめて include されます．これにより，記述の手間を削減し一貫性を保持できます．

● 関数ごとにフレーム（環境）が作られる

各関数は，呼び出し元の関数と独立した環境を持ちます．この環境のことを「フレーム」と呼びます．

プログラム実行中に関数が呼び出されると，**図2**のように新しいフレーム（環境）が作成されます．このフレームの中には，関数の仮引き数や関数の中で宣言されている変数などの各関数ごとの情報が含まれます．

関数はフレーム中の変数にアクセスして処理を行います．呼び出し元の変数に直接アクセスすることはできません．このしくみにより，呼び出し元と呼び出された側で変数名が衝突することは避けられています．また，同じ名前の関数を続けて呼び出しても，それぞれ異なるフレームが作成されます[注3]．

注3：変数の値などが引き継がれないので注意すること．