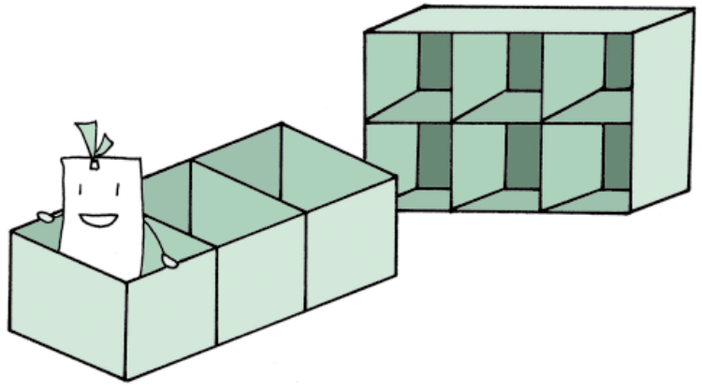
第４章はここがkey

プログラムをより簡潔なものに

この章では**配列**と**ポインタ**について学んでいきます。配列は複数の変数を集めて一列に並べたものです。たとえば、int a[4]; と宣言すると int 型変数 4 個分を表します。この中の 1 つ分の箱（要素）を使うときは a[0]、a[1]、a[2]、a[3] といった 0 からはじまるインデックス番号を指定します。a[i] のように変数をインデックス番号に使えるので、a1、a2、…と変数を増やしていくよりも、宣言や処理を簡単にできます。

また、int a[4] という例では、変数を一列に並べた 1 次元のイメージでしたが、縦横に並べるイメージの 2 次元配列や、3 次元配列、4 次元配列などといったものもでてきます。配列の考え方は、大量のデータを管理する必要があるときは、なくてはならないものといえるでしょう。

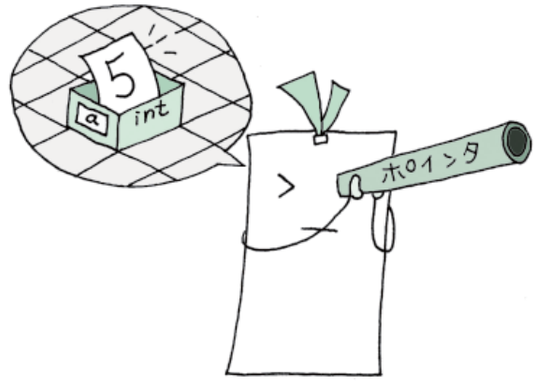


ところで、第 1 章では、「文字列は配列である」と紹介しましたが、この章では配列と文字列の関係をもっと詳しく見ていきます。C 言語には、文字列をコピーしたり、文字数を調べたりできる便利な関数がそろっていますので、それらについても紹介しておきましょう。

ポインタと配列は切っても切れない関係？

ポインタとは、「データのありかを保管しておくための変数」です。それだけ聞くと、なかなかピンときませんが、こんなたとえ話はどうでしょう。あなたは、久しぶりにお気に入りのレストランに行ってみることにしました。しかし、行ってみたらそこにお店はなく、代わりに移転先が書いてある看板があるだけでした。看板の住所に行ってみると、ちゃんとレストランがあり、大好きな料理を食べることができました。このときの看板にあたるのがポインタです。レストランは変数で、料理はその中のデータとすると、ポインタとそれが指し示す変数、そしてその中に入っているデータの関係が少しはイメージできたでしょうか？

実際には、変数やポインタはメモリ上に存在していますので、少しコンピュータの内部のこともお話しする必要があります。メモリには**アドレス**と呼ばれる数字がついていて、メモリ上に存在するデータは、その場所をアドレスで表すことができるのです。それならポインタなんていわずに、アドレスといった方が簡単じゃないの？と思うかもしれませんが、いろいろな種類のコンピュータがある中で、内部のしくみにあまり依存してしまうのはよくありません。また、実はポインタと配列には密接な関係があります。その考え方についてもお話していきます。



「以前 C 言語を勉強してみたけどポインタでつまずいた」という人も「なんだかややこしそうだな」と思っている人も、じっくりと配列とポインタについて学んでいきましょう。

配列

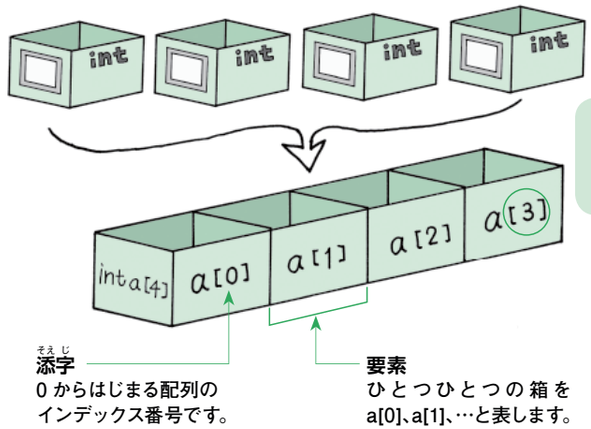
同じ型のデータであれば配列としてまとめて扱うことができます。

配列の考え方

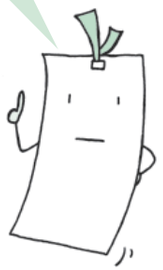
配列は複数の同じ型の変数を 1 つにまとめたものです。大量のデータを扱うときや複数のデータを次々と自動的に読み出したいときは配列を使うと便利です。
配列の宣言は次のように行います。

```
int a[4];
```

型　配列名　配列の大きさ（要素の数）



添字は0からはじまるため、要素数よりも1小さい値になります。



添字（そえじ）
0 からはじまる配列の
インデックス番号です。

要素
ひとつひとつの箱を
a[0]、a[1]、…と表します。

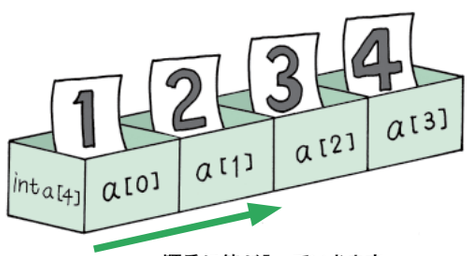
配列を初期化するときは、{} を使って値を列挙します。

```
int a[4] = {1, 2, 3, 4};
```

[] 内の要素数は省略することができます。

```
int a[] = {1, 2, 3, 4};
```

{ } 内にデータがいくつあるかで
要素数を自動的に決定します。



順番に値が入っていきます。

配列要素の参照と代入

配列の要素ひとつひとつは、普通の変数のように参照と代入ができます。

```
int a[4];
int n = 1;

a[0] = 1;
a[1] = 2;
a[2] = 3;
a[3] = 4;
printf("%d¥n", a[n]);
```

a[0] ～ a[3] の値を代入

a[1] の値を表示

添字の数に「0」～「要素数－ 1」以外の値を指定すると、実行時にエラーになるので要注意！

```
int a[4] = {1, 2, 3, 4};
printf("%d", a[4]);
```

a[4]は配列の範囲外なので、プログラムが途中で止まったり、予期しない動きをしてしまいます。

例

```
#include <stdio.h>

main()
{
    int i;
    int a[] = {1, 2, 3, 4};

    for(i = 3; i >= 0; i--)
        printf("%d ", a[i]);
    printf("¥n");
}
```

添字に変数を使って表示を自動化しています。

実行結果

```
4 3 2 1
```