配列について

71 回生　匹田啓吾

こんにちは、こんばんは。71 回生の匹田と申します。まだまだ無知ではありますが、何かと便利な配列について自分なりに解説しようと思います。

1　配列とは？

配列とは、データをまとめて保存できるものです。殆どのプログラミング言語にあり、データ構造の 1 つです。

通常、変数には 1 つのデータしか入れられませんが、複数のデータをまとめて扱うと便利な場合があり、その時に登場するのが、配列です。配列には、多くのデータを入れることができます。これなら、100 個、200 個の変数も 1 つにまとまります。また、変数が見やすくなり、変数を宣言する時間を短縮できます。(また、配列の 1 つ 1 つのデータを要素といいます。)

また、配列はループによって一括して処理することができ、複数のデータの書き換えもとても容易です。

今後の説明のために例えて言うと、データを入れる箱のようなものです。データを保存するためには、箱をデータ 1 つにつき箱を用意しなければいけないのですが、配列を使えば、箱を多く用意しなければならない時、箱を 1 つ 1 つ用意しなければならないところを、まとめてその箱を用意でき、箱をたくさん用意する手間が省けて楽です。とても分かりやすい解説ですね!

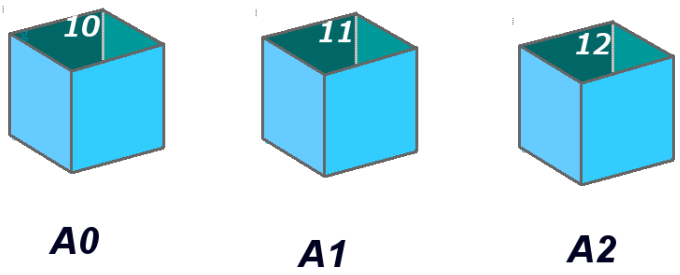


図 1　箱を 1 つ 1 つ。(10,11,12 は要素、A0,A1,A2 は変数名)

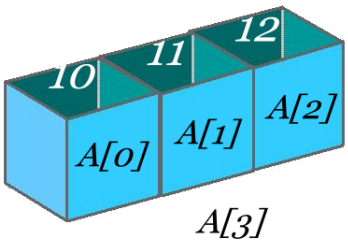


図 2　箱がたくさん。1 つ 1 つ変数を用意する必要がありません！やったね！

しかし、配列の素晴らしいところはこれだけではありません。

2　2 次元配列とは？

ここで、何かと便利な 2 次元配列です。いきなり例をあげます。
下の表に数字を当てはめていくことにします。

1	4	9	16	25	36	49	64	81
100	121	144	169	196	225	256	289	324
324	323	320	315	308	299	288	275	260
243	224	203	180	155	128	99	68	35

全て数を変数に記録します。どのような方法が良いでしょう。

- 選択肢 1:　1 つ 1 つ変数を用意する。
  1 番単純で 1 番面倒くさい方法です。変数はこの場合 36 個用意しなければなりません。出来ないことは無いですが、変数名がごっちゃになる等し易く、より良い方法を探すべきです。
- 選択肢 2:　先程説明した配列で変数をまとめる。
  この方法の長所は、1 つにまとめてある事ですが、短所は「上から 2 番目、左から 5 番目の数はなんでしたか」等表の形に戻す時にややこしくなる事です。

- 選択肢 3:　2 次元配列
  これをしっかり説明していきたいと思います。

3　1 次元配列と 2 次元配列

ここで今まで説明してきた 1 次元配列と新たな配列、2 次元配列について比較したいと思います。

- 1 次元配列

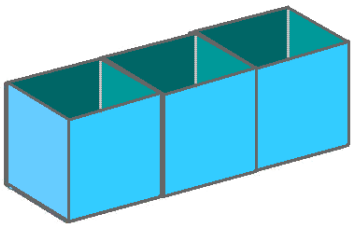


図 3　矢印の向きの 1 方向、直線的に箱 (変数) が設置。

- 2 次元配列

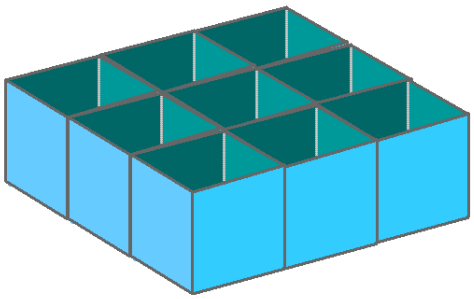


図 4　矢印の向きの 2 方向、平面的に箱 (変数) が設置。

上の図でお分かりいただけたと思いますが、2 次元配列とは、2 方向にデータを配置できる配列です。

上の、表に数字を入れていく例でも、4 × 9 の 2 次元配列を用意するだけで O.K. です。

2 次元配列はマス目があるゲーム (オセロ、将棋、チェスなど) や表の形にできるデータ等、使いどころはたくさんあります。

実は、2 次元配列以上の配列、3 次元配列や、4 次元配列な等まだまだありますが、ここでは、省略することにします。

2 次元配列は、使えるところが決まっていますが、そこでは 1 次元配列よりデータがまとまり扱い易くなる。

1 次元配列は、似たようなデータが集まっているときは便利ですが、一部の状況では 2 次元配列のほうが便利です。

つまり、2 次元配列が使えるのなら使う、といった使い方でよいと思います。

しかし、2 次元配列には 1 次元配列にはない短所があります。それは、一部の言語では使えないということです。それらの言語を使うときには、注意が必要です。

4　配列と添え字

配列の要素のうち先頭から何番目かを番号として指定することができるのが添え字です。配列は、1 次元配列の場合、**配列の名前 [数字]**(下の場合だと A が「配列の名前」3 が数字) で表せる、数字部分を、添え字とよびます。また、要素の数字部分もそう呼びます。

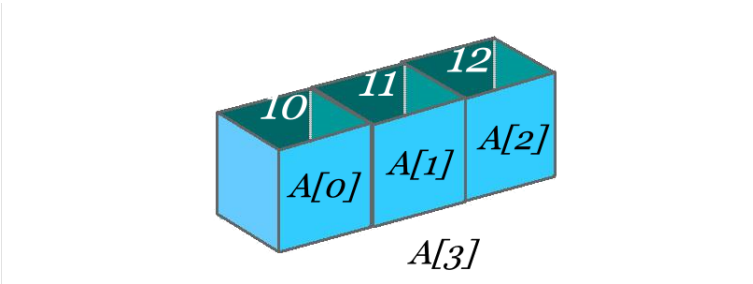


図 5　これが添え字。因みに、C の場合は 0 が配列の先頭の要素を示すことになる。

数字は、配列の先頭から何番目かをしめすもので、大抵のプログラミング言語では、0 から添え字が始まるので、`A[2]` というと、前から 3 番目の要素を指定することになります。

また、2 次元配列の場合だと、`A[4][9]` というような形で表せます。

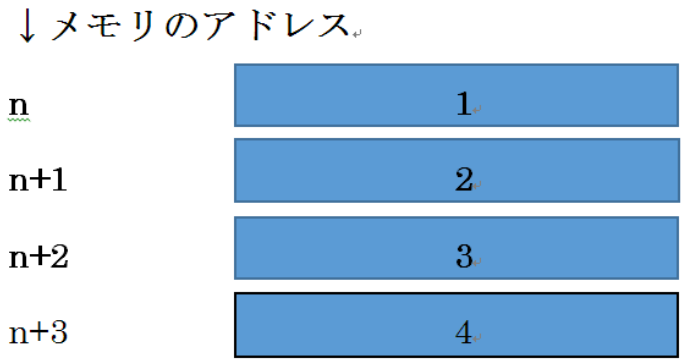
これは、`A[4]` という配列の 9 個の集まり、つまり、配列の配列という形で表していることになります。よって、理論上は 3 次元配列、4 次元配列…と無限に多次元配列はつくれてしまうわけです。

5　添え字が 0 から始まる理由 (C)

ではなぜ添え字は 0 から始まるのでしょう。別の言語には添え字が 1 から始まる言語もあります。

それは、メモリに関係しています。

配列 `A[4]` があり、要素 `A[0]=1,A[1]=2,A[2]=3,A[3]=4` だとすると、



上のようにメモリが確保されます。

つまり、(始めの要素のアドレス)+(添え字) で、その要素のアドレスが参照できる、というわけです。K&R の「プログラミング言語 C 第 2 版」(C の開発者によって著された本) にも

> 「C においては、ポインタ (変数のアドレスを記憶する変数) と配列との間に強い関係がある。この関係は強いので、ポインタと配列を同等に論じなければならない。
> (中略)
> a[i] という記法は、先頭から i 番目の位置の配列要素を参照する。」

つまり、配列はメモリを連続的に確保して、(始めの要素のアドレス)+(添え字)=(その要素のアドレス) となるように作られていたのです。

また、その後 C の影響を受けた多くの言語が同じシステムを採用した、と考えられます。

こうする事の利点として、配列の添え字だけで要素の個数が分かることですが、その程度です。

しかし、数値情報は 2 進数が基本ベースにあると想定しており、`A[256]` 等の時、要素が 255 個しかなく、また 1～256 を 0～255 にする処理があったとしても、256 の分メモリが無駄になってしまいます。

1 から始まる言語は、理解し易さ、計算のし易さ、現実への忠実さ等を優先したものと考えられます。

それぞれがそれぞれの言語に適した方式をとっている、といえます。

6　最後に

　配列について少しでも理解を深めて頂けたでしょうか。そうであれば、とても幸いです。こんな拙い文章を最後まで読んで頂きありがとうございます。