基礎プログラミング演習 II 教材　(#15)

■ 文法落ち穂拾い：include

★教科書 p.70 プリプロセッサ命令（特に図 4.7, 4.8 を参照）

押さえて欲しいポイント：
・プログラムは多くの段階を経てコンパイルされる
・プリプロセッサと呼ばれる前処理機構がある
・#include はそこでヘッダファイルを差し込むものである
・ヘッダファイルには printf( ) などの関数定義があり、それらの利用のために include が必要
・ちなみに #define もプリプロセッサ命令

■ 二次元配列

★教科書 p.107, 6.2 「多次元配列」を参照

実験：サンプルプログラム（p.109 プログラム例 6.6）を取得して実行し、動作を確認せよ。

押さえて欲しいポイント：
・入力の際の scanf() で &table[line][colm] につけられた & は今は気にしない
・二重ループで一つ一つの要素に入力していること
・合計用配列（縦横）には初期化が必要であること
・二重ループで一つずつの要素を処理していること
・こうした動きは図を書いて、紙と鉛筆でトレースして納得すること

最も理解して欲しいこと：
結局のところ、
・C 言語では配列変数（という機能）を用意したが、
・配列をひとつの固まりとして処理するための機能は用意されていない
という構造に注目してください。
そうすれば、
・そのため配列を扱う処理は、必ず一つ一つの要素を個別に処理する必要がある
・多くの場合、これはループによって順繰りに処理されることで実現される
という結果を理解できると思います。

□ 課題 1.

教科書 p.114, 演習問題 6.2 のプログラムを作成してください。

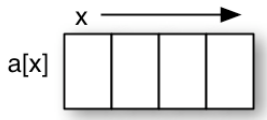
考え方：
・要素を一つずつ処理するしかない
・移動させる要素の行・列の値から、移動先の行・列の値を算出できるはずだ
・算出方法が見えてこない受講生は紙に書いて作業することを勧める

興味のある学生向けに：
対角線移動した結果のデータを保持するために、二次元表（二次元配列）を二つ用意するのではなく、二次元表を二つ内包する一つの三次元配列を用意してみると良い。多次元の配列については次ページに資料を加えておく。

■ 多次元の配列について

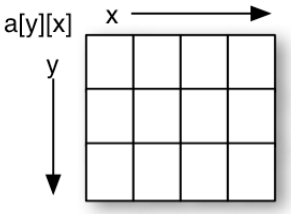
一次元の配列について、例えば右の図のように要素が一列に並んだイメージで考えることが多いでしょう。現在の主流となっているコンピュータのメモリの配置にうまく合う考え方でもあります。

一次元配列 a[4] の例

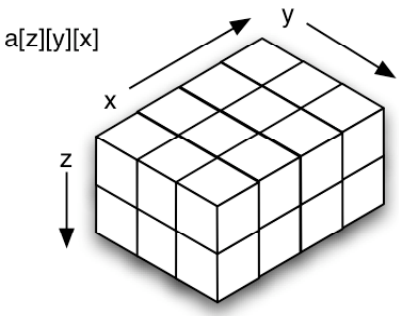


二次元の配列についても下図のように問題なく平面に要素を並べてイメージすることができるとおもいます。三次元の場合も同様に立体でイメージすることができるでしょう。

二次元配列 a[3][4] の例

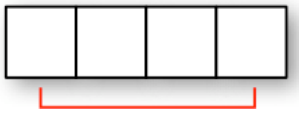


三次元配列 a[2][3][4] の例

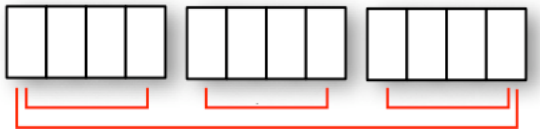


では次元数が更に高次になった場合はどうでしょう。四次元、五次元の配列をうまくイメージすることができますか？

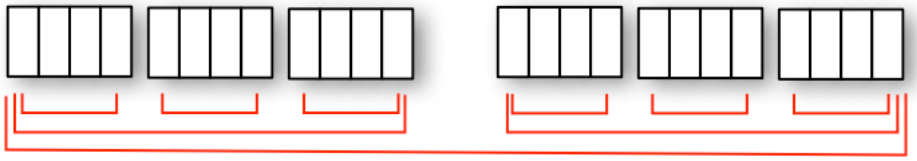
一次元の配列 a[4] はやはり要素が 4 つ並んだものだとしましょう。



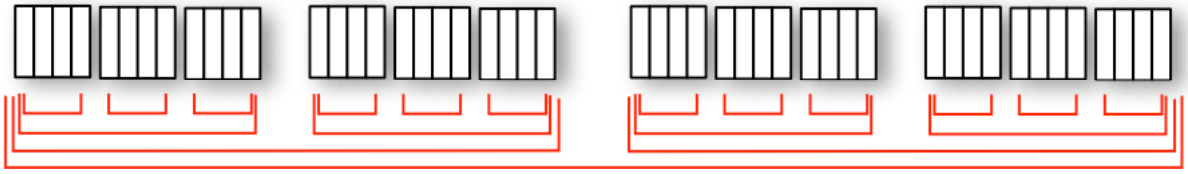
二次元の配列 a[3][4] は、a[4] を 3 セット、更に横に並べたものだと考えてみます。



三次元の配列 a[2][3][4] は、a[3][4] を 2 セット、繰り返して横に並べたものだと考えます。
こうすることで無理なく更により高次元な配列についてもイメージできるのではないかと思います。



例えば四次元の配列ならば下図のようになるでしょう。



実際のアプリケーションで多次元の配列を使う場合は、このように、ある一定のセットを、更に何セットか組み合わせるようなケースが多いでしょう。たとえば一台あたり 10 本の糸巻きがついている糸巻き機が、ある建物では一列に 20 台並んでおり、それが建物あたり 4 列配置されているとします。全 800 本の糸巻きを表す配列変数は spindle[4][20][10] となるでしょう。この建物が一つの工場に 5 つあり、それが 3 つの地域にあるとしたら、配列は自然と spindle[3][5][4][20][10] といった五次元配列となるでしょう。