計算機プログラミングI 講義ノート #8

担当：井上 純一 (情報科学研究科棟 8-13), 赤間 清 (情報基盤センター)
URL：http://chaosweb.complex.eng.hokudai.ac.jp/~j_inoue/PROG2007/PROG2007.html

平成 19 年 6 月 15 日

目 次

18.4 2 次元配列の宣言と初期化

先週学んだ配列は要素数を 1 列に並べた 1 次元配列でしたが, この配列は一般次元に拡張することができます. 例えば, 2 次元の配列は次のように宣言します.

```
/* 第 1 軸が 3，第 2 軸が 2 の要素数 3 x 2 = 6 の 2 次元配列 */

int a[2][3];
```

この初期化は次のように行います.

```
int a[2][3]={10,11,12,13,14,15}; /* 表現 1 */
```

これは次のように書くこともできます.

```
int a[2][3]={{10,11,12},{13,14,15}}; /* 表現 2 */
```

または

```
int a[][3]={10,11,12,13,14,15}; /* 表現 3 */
```

と書けば, 具体的に

```
a[0][0] ← 10
a[0][1] ← 11
a[0][2] ← 12
a[1][0] ← 13
a[1][1] ← 14
a[1][2] ← 15
```

のように初期化されることになります.

表現 2 のように記述すると, 直観的にわかりやすくなると思います. また, 表現 3 のように 2 次元配列 (すぐ後にみる多次元配列でも) では最終軸 (この場合には第 2 軸) のみが省略可能です. 以上をふまえて次の練習問題をやってもらいましょう.

練習問題 18.5

次のプログラムは 2 行 2 列の行列の和を求めるプログラムである. 各自がファイルに入力し, コンパイル・実行することで, その動作を確認せよ.

```
#include<stdio.h>
#include<math.h>

main(){
        double a[2][2]={{4,3},{5,1}},b[2][2]={{1,0},{0,1}},c[2][2];
        int i,j;

        for(i=0;i<2;i++){
                for(j=0;j<2;j++){
                c[i][j]=a[i][j]+b[i][j];
                }
        }
        printf("%lf %lf\n %lf %lf\n",c[0][0],c[0][1],c[1][0],c[1][1]);
 }
```

練習問題 18.6 (※ LMS 入力問題)

次の 2×2 の正方行列:

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix}$$

の積 $\boldsymbol{C} = \boldsymbol{AB}$ を計算し, $\boldsymbol{C}$ の成分を

```
c[0][0]=*** c[0][1]=*** c[1][0]=*** c[1][1]=***
```

の形式で出力するプログラムを作成せよ.

18.5　多次元配列の宣言

配列の次元が例え何次元になろうが, 常に定義でき, 次のように宣言することができます.

```
/* このように何次元でも可能 */

int a[10][23][6].....[8];
```

18.6　配列のアドレス

配列の宣言として, 例えばサイズが 10 であり, その成分が整数型をもつ 1 次元配列 COUNT は

```
int COUNT[10];
```

としましたが, この COUNT という表記は「配列の名前」であると同時にこの配列の先頭, つまり, COUNT[0] のアドレス (記憶媒体上のデータの格納場所) を表しています. 実際に既にみたようにアドレス演算子を変数名の前につけると, 該当するその変数のアドレスを表記するのでしたから

```
&COUNT[0]
```

というのは COUNT[0] のアドレスですが, これは上で説明したように COUNT の値と同じ値を持ちます.

これは多次元の配列の場合も同じであって, 例えば, COUNT2[10][10] において, &COUNT2[0][0] の値と COUNT2 の値は同じになります. このことを確認するための練習問題をやってもらうことにしましょう.

演習問題 18.7

次のプログラムをファイルに書き込み, コンパイル・実行することで結果を確認せよ.

```
#include<stdio.h>

main(){
   int array1[10];
   int array2[10][10];

  /* 配列の先頭アドレスを 16 進数表示する */
    printf("array1=%x, &array1[0]=%x\n",array1,&array1[0]);
    printf("array2=%x, array2[0]=%x, &array2[0][0]=%x\n",array2,array2[0],&array2[0][0]);

}
```

18.7　バブルソートとクイックソート

与えられた数列をその小さい順（あるいは大きい順) に並べ替えるアルゴリズム (ソート) とその C 言語によるプログラミングを学んでいきます. ここでは代表的なソーティングのアルゴリズムである**バブルソート**と**クイックソート**を教科書 pp. 96-99 を参考にして説明していきます.

練習問題 18.8 (※ LMS 入力問題)

バブルソートを実現するプログラムを作成せよ. 具体的には, 0 から 9 までの 10 個の数字をキーボードから入力すると, それを小さい順に並べ替えて

```
**********
```

のように表示せよ.

※ この際, 必要ならば 2 つの変数 a,b を交換する手続き型引数マクロ：

```
#define swap(a,b){int t=(a);(a)=(b);(b)=t;}
```

を用いても良い. この関数マクロは以降頻繁に用いることになる.

練習問題 18.9

クイックソートを実現する次のプログラムをファイルに書き込み, コンパイル・実行し, 動作を確認せよ.
※ プログラムの各関数・各行でどのような処理を行わせているのかをコメントとして書き込むこと.

```
#include<stdio.h>
#define NMAX 10
#define swap(a,b){int t=(a);(a)=(b);(b)=t;}

void print(int array[NMAX], int N)
{
        int i;
        for (i=0; i<N; i++){
             printf("%d", array[i]);
            }
        putchar('\n');
}


void quicksort(int array[NMAX], int lower, int upper)
{
      int bound=array[lower];
      int l=lower;
      int u=upper;

    do{
        while(array[l]<bound) l++;
        while(array[u]>bound) u--;
      if(l<=u){
             swap(array[l],array[u]);
             l++;
             u--;
            }
      } while(l<u);
       if(lower<u) quicksort(array,lower,u);
       if(l<upper) quicksort(array,l,upper);
}
```

```
main()
{
      int array[NMAX];
      int N=0;
        while(1){
            printf("data> ");
            scanf("%d",&array[N]);
          if((array[N]<0) || (++N==NMAX)) break;
}

      quicksort(array,0,N-1);
      print(array,N);
}
```

今週の LMS 入力問題

3×3 の奇数陣を表示するプログラムを作成せよ. 奇数陣とは 1 辺が奇数 n の正方行列に 1 から n^2 までを埋めたとき, 縦, 横, 斜めの n 個の和が全て等しいようなものを言う (ここで考えるのは $n = 3$ の場合).
※ 以下のアルゴリズムを参考にしてもよい (教科書 95 ページの図 6.1 も同時に参照せよ).

奇数陣作成アルゴリズム

(1) n 次正方行列を用意する. 「注目要素 a」を最下段中央とし, $a = 1$ とする.
(2) 注目要素に数 a を置き, それを以下のルールで変更する.

> **注目要素の更新規則：**
> 斜め右下に既に数が埋まっている場合には注目要素の上の要素を, また, 空いている場合には斜め右下の要素を新しい注目要素とする. 注目要素が行列外の場合には, 周期的に配置されているものと考える.

(3) a を 1 だけ増やし, 行列が埋まるまで (2) を繰り返す.

ただし, 出力結果を

```
* * *
* * *
* * *
```

の形式で表示せよ.

(参考までに)

皆さんの中には得られる奇数陣がユニークかどうかが気になっている人もいるかもしれません. 出来上がる 3×3 の奇数陣としては

4	9	2
3	5	7
8	1	6

か, この回転対称の配置しかありえません. この事実は次のようにして示すことができます. まず, 各マスに入る数字を形式的に

$X_{0,0}$	$X_{1,0}$	$X_{2,0}$
$X_{0,1}$	$X_{1,1}$	$X_{1,2}$
$X_{0,2}$	$X_{1,2}$	$X_{2,2}$

のように行列成分として表すことにすると, 奇数陣の要請より

$$X_{0,1} + X_{1,0} + X_{2,0} = a \tag{12}$$

$$X_{0,1} + X_{1,1} + X_{2,1} = a \tag{13}$$

$$X_{0,2} + X_{1,2} + X_{2,2} = a \tag{14}$$

となることが必要であり, 全ての数字の和は 45 になるべきですから

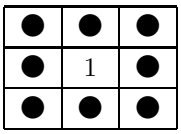
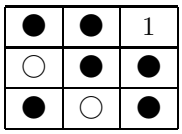
$$\sum_{j=0}^{2}\sum_{i=0}^{2} X_{i,j} = \sum_{i=1}^{9} i = 45 \tag{15}$$

であり, (12)-(14) を (15) 式に代入すると

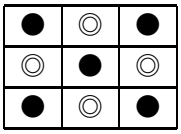
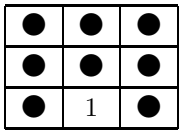
$$a = 15 \tag{16}$$

が得られます. 従って, 縦, 横, 斜めの数字の和はいずれも 15 でなければならないことがわかります. 従って, これからはこの条件下で各数字を配置していくことを考えます.

まずは数字の 1 をどこに置くかを考えます. 縦, 横, 斜めの 3 つの数字を足しあげると 15 にならなければならないわけですから, 1 が置かれた同じ縦, 横, 斜めの列に $2, 3, 4$ が来ることはできません. 例えば, 1 と 4 が同じ列に来たとすると, $15 - (1 + 4) = 10$ となりますから, 残る一つの場所に今存在しない 10 を置かなければならなくなるからです. 従って, 次のような箇所に 1 を置くことはできないことになります.

上の左側の例では 1 が中央にありますから, $2, 3, 4$ をどのように配置しょうが, 必ず, 1 と同じ列になってしまいます. 一方, 右側の例では, 1 と同じ列にならない場所は○の 2 箇所のみであり, $2, 3, 4$ のいずれか 1 つの置き場所がなくなります. 従って, 結局, 1 の置ける場所としては

上図の左側に示したように◎の箇所のみとなり, 以下では対称性を考えて, 上の図の右側に示した場所に 1 を置くことにします.

次に考えるのは 2, 3 の置く場所ですが, 2, 3 が同じ列に来ることもできません. なぜならば, 同じ列に 2, 3 があれば, 同じ列の残りの 1 箇所には $15-(2+3)=10$ を置かなくてはならないからです. このことを考えると 2, 3 の配置の仕方としては次のものとその回転対称なものに限られることがわかります.

B	●	2
3	●	A
●	1	●

次に 4 を置くことを考えると, 4 は 1 と同じ列に来てはいけませんでしたから, 上図の A,B の 2 箇所のみとなります. まず, A に 4 が来たとすると, 各列の数字の和が 15 となることから, 残りの数字の置き方は自然に決まって行き, 順調に最後まで行きそうに思えるのですが

7	6	2
3	8	9
5	1	9

が最終的に得られてしまい, 斜めの 1 列が $7+8+9=24$ となり, 15 を超えてしまいアウトです. 従って, 4 の置き方としては前図の B の場所しかなく, この場合に他の数字を並べていけば, 結局

4	9	2
3	5	7
8	1	6

が得られます. 従って, 可能な 9 つの数字の配置はこれか, あるいは, この回転対称なものに限られることがわかります.