解答編　第 6 章

多数データの取り扱い

演習問題 6.1　文法事項

1)「int number[10]; 」と配列宣言した場合、最大の要素番号はいくつか答えなさい。

解答例　最大の要素番号は 9

C 言語の配列の配列番号は 0 から始まります。このため、最大の要素番号は、宣言した要素数よりも 1 だけ小さな値になることに注意して下さい。

2) 1) のように宣言した配列で、最大の要素番号より大きな要素番号を持つ配列要素にデータを代入した場合、プログラムの動作の可能性について論じなさい。

解答例　問題に示されたような状況の場合、プログラムの動作は不定となります。最大の要素番号より大きな要素番号を持つ配列要素に代入を行うと、宣言により確保された配列の領域外のメモリにデータを格納することになりますが、運が良ければ代入は成功します。しかし、多くの場合は、プログラムが暴走や異常停止します。

このことは、プログラムのデバッグに際して大きな問題になります。いつでも、必ず問題を起こすバグを見つけるのは比較的簡単ですが、動いたり動かなかったりと動作が一定しないバグは発見が困難です。場合によっては、(見かけ上) 常に正常に動作し、バグが発見されないという可能性もあります。そのようなプログラムは、常に爆弾を抱えて動作しているようなものです。

演習問題 6.2

要素が整数からなる３×３の表データを読み込み、対角線を軸にして要素を入れ替えた表を表示するプログラムを作りなさい。例えば

```
1 2 3                1 4 7    9 6 3
4 5 6 という表に対して 2 5 8 と 8 5 2 を出力します。
7 8 9                3 6 9    7 4 1
```

解答例　この問題は、配列の添字番号を自由に操作する練習です。解答例を解答プログラム 6.1 に示します。

慣れていないと、2 番目の対角線入れ替えで、かなり混乱して頭を悩ませたかもしれません。これが、要素数が増えたり、次元が増えたりすると、もっと大変です。多次元配列の模式図を紙の上や頭の中にしっかり描いて考えるようにしましょう。多次元配列で混乱した場合には、簡単な数値例でプログラムを追ってみることが、意外と早い解決に導いてくれることもあります。

解答プログラム 6.1　行列の要素入れ替え表示プログラム

```
/************************************************
    p6_1.c
    行列の要素入れ替え表示プログラム
    Tadaaki Shimizu    2006.11.14
************************************************/
#include <stdio.h>

main()
{
    int matrix[3][3]; // 3×3 行列
    int i, j;         // カウンタ変数

    // 行列の入力
    for(i = 0; i < 3; i = i + 1) {
        for(j = 0; j < 3; j = j + 1) {
            printf("[%d][%d] 要素を入力 > ", i, j);
            scanf("%d", &matrix[i][j]);
        }
    }

    // 行列表示
    printf(" 入力された行列 \n");
    for(i = 0; i < 3; i = i + 1) {
        for(j = 0; j < 3; j = j + 1) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
    printf(" 対角線反転 \n");
    for(i = 0; i < 3; i = i + 1) {
        for(j = 0; j < 3; j = j + 1) {
            printf("%d ", matrix[j][i]);
        }
        printf("\n");
    }
    printf(" 対角線反転 2\n");
    for(i = 2; i >= 0; i = i - 1) {
        for(j = 2; j >= 0; j = j - 1) {
            printf("%d ", matrix[j][i]);
        }
        printf("\n");
    }

    exit(0);
}
```

演習問題 6.3

1）３×３の行列を２つ読み込んで、その和を求めるプログラムを作りなさい。

解答例　解答例を解答プログラム 6.2 に示します。

行列の和は非常に簡単な操作であるため、プログラム中で和を行っている部分は、ほんの５行に過ぎません。プログラムの大部分がデータの入出力にあてられています。

このプログラムでは、行列の入力部分などを簡単にするために、3×3 の２次元行列のために、３次元の配列を用いています。プログラムをよく読んで、３次元配列がどのように用いられているか理解して下さい。

先にも述べたように、行列の足し算は非常に単純な操作です。その上、「行列の入力」の部分と、「行列の和」の部分をよく観察すると、行列を入力しながら同時に和を計算することが可能であることに気付きます。そのようにプログラムを設計すると、`for` 文による繰り返しを減らすことができ、プログラムが短くなり、実行の上でも無駄がありません。しかし、そのような方法は多くの場合、良い方法ではありません。プログラムは、一般的に、「データの入力」「データの処理」「結果の出力」の３つの部分から成ります。後のプログラムのメンテナンスなどを考えると、これらの機能が異なる部分は独立させて設計しておく方が良いのです。

このように、プログラムの中身を機能別に整理して書いておくと、別の便利さもあります。例えば、次の 2) の問題は、解答プログラム 6.2 をほとんど流用して実現できます。「行列の和」の部分を「行列の積」に書き換えるだけで、他の部分はそのまま使えるからです。

「結果の出力」の部分は、面倒なので、ちょっと手を抜いてあります。もう少しマシな出力表示になるように、各自工夫してみて下さい。

2）３×３の行列を２つ読み込んで、その積を求めるプログラムを作りなさい。

解答例　解答例を解答プログラム 6.3 に示します。

解答プログラム 6.2 とよく見比べて下さい。

解答プログラム 6.2　行列の和を求めるプログラム

```
/************************************************
    p6_2.c
    行列の和プログラム
    Tadaaki Shimizu    2006.11.14
************************************************/
#include <stdio.h>

main()
{
    int inputMatrix[2][3][3]; // 入力用３×３行列 ２つ
    int answer[3][3];         // 和行列
    int i, j, k;              // カウンタ変数

    // 行列の入力
    for(i = 0; i < 2; i = i + 1) {
        printf("<< %d 番目の行列入力 >>\n", i);
        for(j = 0; j < 3; j = j + 1) {
            for(k = 0; k < 3; k = k + 1) {
                printf("  [%d][%d] 要素を入力 > ", j, k);
                scanf("%d", &inputMatrix[i][j][k]);
            }
        }
    }

    // 行列の和
    for(i = 0; i < 3; i = i + 1) {
        for(j = 0; j < 3; j = j + 1) {
            answer[i][j] = inputMatrix[0][i][j] + inputMatrix[1][i][j];
        }
    }

    // 結果の出力
    for(i = 0; i < 2; i = i + 1) {
        for(j = 0; j < 3; j = j + 1) {
            for(k = 0; k < 3; k = k + 1) {
                printf("%d ", inputMatrix[i][j][k]);
            }
            printf("\n");
        }
        if(i == 0) printf("  と\n");
    }
    printf("  を足すと\n");
    for(i = 0; i < 3; i = i + 1) {
        for(j = 0; j < 3; j = j + 1) {
            printf("%d ", answer[i][j]);
        }
        printf("\n");
    }
    exit(0);
}
```

プログラム例 6.3　**行列の積を求めるプログラム**

```
/************************************************
    p6_3.c
    行列の積プログラム
    Tadaaki Shimizu    2006.11.15
************************************************/
#include <stdio.h>

main()
{
    int inputMatrix[2][3][3]; // 入力用３×３行列 ２つ
    int answer[3][3];         // 和行列
    int i, j, k;              // カウンタ変数

    // 行列の入力
    for(i = 0; i < 2; i = i + 1) {
        printf("<< %d 番目の行列入力 >>\n", i);
        for(j = 0; j < 3; j = j + 1) {
            for(k = 0; k < 3; k = k + 1) {
                printf("  [%d][%d] 要素を入力 > ", j, k);
                scanf("%d", &inputMatrix[i][j][k]);
            }
        }
    }

    // 行列の積
    for(i = 0; i < 3; i = i + 1) {
        for(j = 0; j < 3; j = j + 1) {
            answer[i][j] = 0;
            for(k = 0; k < 3; k = k + 1) {
                answer[i][j] = answer[i][j] +
                          inputMatrix[0][i][k] * inputMatrix[1][k][j];
            }
        }
    }

    // 結果の出力
    for(i = 0; i < 2; i = i + 1) {
        for(j = 0; j < 3; j = j + 1) {
            for(k = 0; k < 3; k = k + 1) {
                printf("%d ", inputMatrix[i][j][k]);
            }
            printf("\n");
        }
        if(i == 0) printf("  と\n");
    }
    printf("  を足すと\n");
    for(i = 0; i < 3; i = i + 1) {
        for(j = 0; j < 3; j = j + 1) {
            printf("%d ", answer[i][j]);
        }
        printf("\n");
    }
    exit(0);
}
```

演習問題 6.4

100以下の2つの整数 x と y を入力し、100以下で、xで割れない数、yで割れない数、両方で割れない数を表示するプログラムを作りなさい。

?例題　この問題は、全く問題のための問題という趣きで申し訳ありません。

2 つの整数で割れない数を見つける程度なら、普通に数を 1 つずつ割り算チェックをする方が良いかもしれませんが、ここではエラトステネスのふるいに類似した方法でプログラムを作ることにしましょう。

解答例を解答プログラム 6.4 に示します。このプログラムでは、エラトステネスのふるいと同じように、数をチェックするための配列 **number[]** を用いています。2 つの整数 ***x*** と ***y*** で割り切れるかどうかをチェックしなければならないため、チェックのための定数として、**X_DIVISIBLE(1)** と **Y_DIVISIBLE(2)** を使っています。チェックの入れ方は、エラトステネスのふるいと全く同じ考え方です。

チェックが終わった後では、

1) **number[i]** の値が **0** なら ***x*** と ***y*** 両方で割り切れない。
2) **number[i]** の値が **1** なら ***x*** で割り切れる。
3) **number[i]** の値が **2** なら ***y*** で割り切れる。
4) **number[i]** の値が **3** なら ***x*** と ***y*** 両方で割り切れる。

となっています。

結果の表示では、これを利用していますが、プログラムの意図がよく読み取れない記述になっています。

このように、整数値を使ってデータを分類するような問題では、データのビットを操作するビット演算子を用いると、より明確なプログラムを書くことができます。本書では紙幅の関係で、省きましたが、興味がある方は、是非勉強してみて下さい。

解答プログラム 6.4　割り切れない数を求めるプログラム

```
/************************************************
    p6_4.c
    割れない数
    Tadaaki Shimizu    2006.11.15
************************************************/
#include <stdio.h>
#define MAX             100 // 素数チェックの最大値
#define NOT_DIVISIBLE   0   // 割り切れない
#define X_DIVISIBLE     1   // xで割り切れる
#define Y_DIVISIBLE     2   // yで割り切れる

main()
{
    int number[MAX+1];        // チェック配列
    int xDivisor, yDivisor;   // 割る数
    int i, j;                 // カウンタ変数

    // 2つの除数の入力
    printf(" xの入力 : ");
    scanf("%d", &xDivisor);
    printf(" yの入力 : ");
    scanf("%d", &yDivisor);

    // チェック配列の初期化
    for(i = 1; i <= MAX; i = i + 1)
        number[i] = NOT_DIVISIBLE;

    // 割り切れるかチェック
    for(i = xDivisor; i <= MAX; i = i + xDivisor)
        number[i] = number[i] + X_DIVISIBLE;
    for(i = yDivisor; i <= MAX; i = i + yDivisor)
        number[i] = number[i] + Y_DIVISIBLE;

    // 結果の表示
    printf("[x = %dで割り切れない数]\n", xDivisor);
    for(i = 1; i <= MAX; i = i + 1)
        if(number[i] % Y_DIVISIBLE == 0) printf(" %d,", i);
    printf("\n");
    printf("[y = %dで割り切れない数]\n", yDivisor);
    for(i = 1; i <= MAX; i = i + 1)
        if(number[i] / Y_DIVISIBLE == 0) printf(" %d,", i);
    printf("\n");
    printf("[両方で割り切れない数]\n");
    for(i = 1; i <= MAX; i = i + 1)
        if(number[i] == 0) printf(" %d,", i);
    printf("\n");

    exit(0);
}
```