# 繰り返しの処理

### *演習問題 5.1　文法事項*

1）while 文、do ～ while 文、for 文の 3 つの繰り返し構文について、その特徴を述べなさい。またプログラム中での使い分けはどのようにしますか。

**解答例**　**wihile** 文は、条件前置型の汎用的な繰り返し構文です。

**do ～ while** 文は、条件後置型の汎用的な繰り返し構文です。

**for** 文は、条件前置型で、回数指定型の繰り返し構文です。

一般に、回数指定型の繰り返しの場合は **for** 文を用います。それ以外の一般的な繰り返しの場合は、条件を繰り返しのどの箇所で判断するかによって、**while** 文と **do ～ while** 文を使い分けます。

2）無限の繰り返しを作る方法にはどのような方法がありますか。

**解答例**　次のように、**while** 文の条件部を定数の **1** とする方法と、

```
while(1) 繰り返し処理 ;
```

**for** 文の第 2 フィールド ( 繰り返し条件 : 終値 ) を空にする下記の方法があります。

```
for(;;) 繰り返し処理 ;
```

**while** 文で無限の繰り返しを作る場合、条件部に定数 **1** を直接書く代わりに、**#define** 命令を使い、**FOREVER** などの定数名を用いて記述すると、読みやすくなります。

3）繰り返しに従属する処理の途中で、繰り返しの条件を判定する形の繰り返しを作る方法について説明しなさい。

**解答例**　前の設問の無限の繰り返しを作り、その中で **if** 文と **break** 文によって、繰り返しを破ることで、条件を処理の途中で判断する型の繰り返しを作ります。具体的には、下記のような構造になります。

```
for(;;) { // 無限ループ
    処理 A;
    if( 条件 ) break;
    処理 B;
}
```

または、

```
while(1) { // 無限ループ
    処理 A;
    if( 条件 ) break;
    処理 B;
}
```

### *演習問題 5.2*

1）プログラム中に掛け算の演算子を用いずに、2 整数の掛け算を行うプログラムを作りなさい。

**解答例**　掛け算の演算子が禁じ手なので、被乗数を乗数回足し合わせることで、積を計算します。負の値にも対応できるように、符号の判定も行ったプログラムを、解答プログラム 5.1 に示します。結果を確かめられるように、掛け算の演算子を使った計算結果を正解として出力してあります。

**解答プログラム 5.1　掛け算プログラム**

```
/************************************************
    p5_1.c
    掛け算プログラム
    Tadaaki Shimizu    2006.11.10
************************************************/
#include <stdio.h>

main()
{
    int multiplicand;  // 被乗数
    int multiplier;    // 乗数
    int product;       // 積
    int sign;          // 符号
    int i;             // カウンタ変数

    // 数値入力
    printf("掛けられる数 : ");
    scanf("%d", &multiplicand);
    printf("掛ける数 : ");
    scanf("%d", &multiplier);

    // 正解出力
    product = multiplicand * multiplier;
    printf("MULT : %d\n", product);

    // 符号の判定
    if(multiplier < 0) {
        multiplier = -multiplier;
        sign = -1;
    }
    else {
        sign = 1;
    }

    // 掛け算の処理
    product = 0;
    for(i = 0; i < multiplier; i++)
        product = product + multiplicand;
    if(sign < 0) product = - product;

    // 演算結果の出力
    printf("MULT : %d\n", product);
    exit(0);
}
```

2）プログラム中に割り算の演算子と剰余の演算子を用いずに、2 整数の割り算の商と余りを求めるプログラムを作りなさい。

**解答例** 1) の問題と同じような問題です。しかし、掛け算よりも割り算の方がちょっとだけ厄介です。割り算の結果が負の場合、商は、0 の方向に丸めることにしました。例えば、割り算の結果が -1.5 なら、商は -1 です。これは、新しい C99 の規格で定められた割り算の演算子の振る舞いと同じです。

1) の問題と同じように、符号の扱いと割り算の操作を分けて考える方がスマートにプログラムを書けます。しかし、同じようなプログラムでは面白くないので、被除数と除数の符号判定で、5 つの場合に分けて、それぞれ割り算処理をするようにプログラムを作ってみました。

この例に限らず、同じ働きをするプログラムを、様々な方法で書くことができます。一つの課題に対して、違った方法でいくつもプログラムを作ってみるというのも、良い勉強になるでしょう。

**解答プログラム 5.2　割り算プログラム**

```
/************************************************
    p5_2.c
    割り算プログラム
    Tadaaki Shimizu    2006.11.10
************************************************/
#include <stdio.h>

main()
{
    int dividend;  // 被除数
    int divisor;   // 除数
    int quotient;  // 商
    int remainder; // 余り

    // 数値入力
    printf("割られる数 : ");
    scanf("%d", &dividend);
    printf("割る数 : ");
    scanf("%d", &divisor);

    // 正解出力
    quotient = dividend / divisor;
    remainder = dividend % divisor;
    printf("DIV : %d, MOD : %d\n", quotient, remainder);

    // 割り算処理
    if(dividend >= 0 && divisor > 0) {
        remainder = dividend;
        for(quotient = 0; remainder > divisor; quotient++)
            remainder = remainder - divisor;
    }
```

```
    else if(dividend >= 0 && divisor < 0) {
        remainder = dividend;
        divisor = -divisor;
        for(quotient = 0; remainder > divisor; quotient++)
            remainder = remainder - divisor;
        quotient = -quotient;
    }
    else if(dividend < 0 && divisor < 0) {
        remainder = dividend;
        for(quotient = 0; remainder < divisor; quotient++)
            remainder = remainder - divisor;
    }
    else if(dividend < 0 && divisor > 0) {
        remainder = dividend;
        divisor = -divisor;
        for(quotient = 0; remainder < divisor; quotient++)
            remainder = remainder - divisor;
        quotient = -quotient;
    }
    else {
        printf("ERROR: 0 での割り算は出来ません。\n");
        exit(0);
    }

    // 結果の出力
    printf("DIV : %d, MOD : %d\n", quotient, remainder);
    exit(0);
}
```

### *演習問題 5.3*

プログラム例5.8を参考にして、九九の表を表示するプログラムを作りなさい。

**解答例**　この問題のプログラムは、**for** 文の入れ子構造を用いると簡単に作ることができます。解答例を解答プログラム 5.3 に示します。

解答プログラム 5.3 では、九九の表の行と桁を数えるために、変数 **line** と変数 **column** を用いています。この 2 つの変数をカウンタ変数として、入れ子構造になった 2 つの **for** 文を使って、**product = line * column** を計算することで、九九の表の中の数値を計算しています。

このような問題では、処理結果の表示方法も重要になります。実行例では、数値がキチンと並んで表示されています。このために、**printf()** 関数の書式文字列の中で、「**%2d**」という書式文字を用いています。これは、10 進整数を 2 桁で表示する指定ですが、詳しくは、第 8 章を読んで下さい。

**解答プログラム 5.3**　**九九の表を作るプログラム**

```
/************************************************
    p5_3.c
    九九の表を作るプログラム
    Tadaaki Shimizu    2006.11.13
************************************************/
#include <stdio.h>

main()
{
    int line;     // 行
    int column;   // 桁
    int product;  // 積

    // 九九の表を作る
    printf("   |  1  2  3  4  5  6  7  8  9\n");
    printf("---------------------------------\n");
    for(line = 1; line < 10; line = line + 1) {
        printf("%2d |", line);
        for(column = 1; column < 10; column = column + 1) {
            product = line * column;
            printf(" %2d", product);
        }
        printf("\n");
    }
    exit(0);
}
```

**[実行例]**

```
% ./p5_3
   |  1  2  3  4  5  6  7  8  9
---------------------------------
 1 |  1  2  3  4  5  6  7  8  9
 2 |  2  4  6  8 10 12 14 16 18
 3 |  3  6  9 12 15 18 21 24 27
 4 |  4  8 12 16 20 24 28 32 36
 5 |  5 10 15 20 25 30 35 40 45
 6 |  6 12 18 24 30 36 42 48 54
 7 |  7 14 21 28 35 42 49 56 63
 8 |  8 16 24 32 40 48 56 64 72
 9 |  9 18 27 36 45 54 63 72 81
%
```

### *演習問題 5.4*

次のゲーム・プログラムを作りなさい。

1）奇数枚のコインを用意します。

2）2 人のプレーヤーが交互に、1 から 4 枚の範囲で好きな枚数のコインをとります。

3）コインが無くなったとき、偶数枚のコインを持っていた方が勝ちです。

プログラムの仕様はおおむね次のようにしなさい。

1）コインの総数を入力してゲームを始める

2）コンピュータは、コインの残り数と、先攻と後攻のコインの持ち数、次にコインをとるのは先攻か後攻のどちらかを表示し、次に取るコイン枚数を入力させます。

3）コインが無くなるまで、2 ) を繰り返します。

4）コインが無くなると、先攻と後攻のコインの持ち数と勝敗を表示してプログラムを終了します。

**解答例**　ユーザーからの入力を沢山必要とするこの手のプログラムは、一般的な数値計算のプログラムに比べて、非常に面倒の多いものです。プログラムの内容が簡単なものでも、ユーザーの入力の便宜を図ったり、入力のエラーチェックをするために非常に煩雑なプログラムになってしまいます。

解答例として示した解答プログラム 5.4 では、最低限のエラーチェックとその処理だけで済ませています。コイン総数の入力では、1 より小さな数や偶数が入力されると、プログラムを終了するようにしています。また、各ターンのコイン取り枚数の入力では、可能な範囲を外れた入力をすると、再度入力を促すようにしています。

実は、解答プログラム 5.4 のコイン取り枚数の入力で行っているエラー処理の方法は、良い方法ではありません。ユーザーに正しい入力を強制する場合に、「正しい入力がなされるまで入力処理を抜け出せない」というやり方は、ユーザーを大変苛立たせます。せめて、入力をあきらめてキャンセルするという逃げ道を作っておくべきです。とは言え、今は C 言語を学んでいる最中ですから、簡単に実現できる方法として、このような方法を採用しておきました。

もう一つ、ゲームを繰り返す際に、先攻と後攻を区別するために、変数 **turnFlag** を用いています。このように何かを区別するための印の役割りをする変数をフラグと呼びます。変数 **turnFlag** の値が **FIRST(0)** なら先攻、**SECOND(1)** なら後攻です。各ターンの最後に、

```
turnFlag = (turnFlag + 1) % 2;
```

を計算しています。この計算をすることで、もし **turnFlag** が **0** ならば **1** に、**turnFlug** が **1** ならば **0** になり、次のターンの攻めを切り替えることができます。このような剰余演算の利用は、プログラミングの定石としてよく用いられるので、この際に覚えておくとよいでしょう。

また、先攻と後攻でほとんど同じ処理が書かれているのも気になります。次の第 6 章で配列を学べば、このプログラムはもっとスマートに書くことができます。

**解答プログラム 5.4　コイン取りゲーム**

```
/************************************************
    p5_4.c
    コイン取りゲーム
    Tadaaki Shimizu    2006.11.13
************************************************/
#include <stdio.h>

#define MIN    1  // コインの最小取り数
#define MAX    4  // コインの最大取り数
#define FIRST  0  // 先攻
#define SECOND 1  // 後攻

main()
{
    int total;           // コインの総数
    int firstPlayer;     // 先攻
    int secondPlayer;    // 後攻
    int remainder;       // コインの残り
    int turnFlag;        // 先攻／後攻のフラグ
    int coin;            // 取るコインの数

    // コインの総数入力
    printf("コインの総数 : ");
    scanf("%d", &total);
    if(total < 1 || total % 2 == 0) {
        printf("ERROR: コインの総数は１以上の奇数にして下さい。\n");
        exit(0);
    }

    // ゲーム開始
    turnFlag = FIRST;
    remainder = total;
    firstPlayer = secondPlayer = 0;
    do {
        printf("コイン残り : %d, 先攻 : %d, 後攻 : %d\n",
                remainder, firstPlayer, secondPlayer);
        if(turnFlag == FIRST) {
            do {
                printf("先攻：コイン枚数 > ");
                scanf("%d", &coin);
            } while(coin < MIN || coin > MAX || coin > remainder);
            remainder = remainder - coin;
            firstPlayer = firstPlayer + coin;
        }
        else {
            do {
                printf("後攻：コイン枚数 > ");
                scanf("%d", &coin);
            } while(coin < MIN || coin > MAX || coin > remainder);
            remainder = remainder - coin;
            secondPlayer = secondPlayer + coin;
        }
        turnFlag = (turnFlag + 1) % 2;
    } while(remainder > 0);// ゲームの終了判定  コインが無くなったら終わり
```

```
    // ゲーム終了　勝敗判定
    printf("コイン残り : %d, 先攻 : %d, 後攻 : %d\n",
                remainder, firstPlayer, secondPlayer);
    if(firstPlayer % 2 == 0 && secondPlayer % 2 != 0)
        printf("先攻の勝ち\n");
    else if(firstPlayer % 2 != 0 && secondPlayer % 2 == 0)
        printf("後攻の勝ち\n");
    else
        printf("ERROR: 予期せぬエラーです。\n");

    exit(0);
}
```