解答編　第4章

条件による分岐2

演習問題 4.1　文法事項

1）条件分岐のために switch ～ case 文が使用できるのは、条件がどのような形で表される場合か答えなさい。

解答例　分岐条件が、整数の一致条件で表されている場合に用いることができます。

2）本文中で説明した、switch ～ case 文を使う場合の注意事項を 2 つ答えなさい。

解答例　・**switch ～ case** 文は原則として **break** 文との組み合わせで使うこと。
・**default** 文を必ず付けること。

3）マジック・ナンバーとは何か答えなさい。また、マジック・ナンバーがプログラムに及ぼす悪影響とは何か答えなさい。

解答例　マジック・ナンバーとは、プログラム中に直接書き込まれた定数のことです。マジック・ナンバーは、見ただけでは、それが何であるのか、あるいは、どんな役割りを果たすのかがわかりません。このため、マジック・ナンバーが書き込まれたプログラムは読んで理解することが困難になります。

4）マジック・ナンバーをなくすために、どのような方法があるか答えなさい。

解答例　マジック・ナンバーを無くすためには、**#define** 命令を用いた定数定義を用いることができます。例えば、足し算を表すコード番号として **1** を使いたい場合、

```
#define ADD 1
```

のように定義することで、直接的な意味が無い **1** という定数でなく、**ADD** という意味を読み取るとこができる定数名を使ってプログラムを書くことができ、プログラムを読みやすくすることができます。

［高度な話題］
C 言語の新しい規格では、内容を書き換えることができない **const** 変数が使えるようになっています。この変数は名前の通り、定数のために用いることができます。例えば、

```
const int ADD = 1;
```

のように **const** 変数 **ADD** を初期化付きで宣言すると、**#define** 命令と同じように用いることができます。#define 命令がプリプロセッサの命令であるのと異なり、**const** 変数は C 言語自体の文法に含まれるため、こちらの方を好んで用いるプログラマも多くいます。
さらに、この例のような定数には、本文の第 15 章の列挙型を用いるのが、最も良い方法です。

演習問題 4.2

整数値を入力すると、その整数が偶数か奇数かを答えるプログラムを作りなさい。ただし、

1）if ～ else 文を用いて作ること。

解答例　この問題は、本当に簡単です。しかし、C 言語のとんでもない落とし穴を含んでいます。まず、解答プログラム 4.1 を示しましょう。このプログラムは、整数を 2 で割った余りが 0 なら偶数という性質を用いています。ちゃんと動きますよ。

解答プログラム 4.1　偶数／奇数判定プログラム (if ～ else 版)

```
/************************************************
    p4_1.c
    偶数／奇数判定プログラム（if ～ else 版）
    Tadaaki Shimizu    2006.11.10
************************************************/
#include <stdio.h>

main()
{
    int inputNumber;    // 入力される整数

    // データの入力
    printf("Input Integer Number > ");
    scanf("%d", &inputNumber);

    // 偶数／奇数判定
    if(inputNumber % 2 == 0)
        printf("%d は偶数です。\n", inputNumber);
    else
        printf("%d は奇数です。\n", inputNumber);
    exit(0);
}
```

2）switch ～ case 文を用いて作ること。

解答例　偶数／奇数判定プログラムを **switch** ～ **case** 文で作った例を、解答プログラム 4.2 に示します。**switch** ～ **case** 文の使用上の注意通り、**break** 文との組み合わせで使っています。さらに、必要無いと思われる **default** 文も付けて、エラーメッセージを出すようにしました。しかし、**2** で割った余りは、**0** か **1** なので **default** 文は、絶対に実行されないのではないでしょうか。

意外なのは、実行例です。**-1** の入力に対して「異常です。」というメッセージが表示されました。**default** 文が実行された証拠です。

実は、負の数の割り算で商と余りを求めるのは意外と問題含みなのです。例えば、**–1** を **2** で割ると、**–0.5** です。このとき、「商は **–1** で余りは **1**」でしょうか？それとも、「商は **0** で余りは **–1**」でしょうか？　要は、**-0.5** を **–1** に丸めるか、**0** に丸めるかの問題です。

C 言語の規格では、この問題は規定されていませんでした。つまり、コンパイラやマシンによって結果はバラバラだったということです。つまり、割り算を用いたプ

ログラムには、移植性がありません。これでは困るので、C99 の規格では、割り算の丸めを、0 の側にするように定められました。つまり、先ほどの計算結果は、「商は 0 で余りは -1」です。

結局、解答プログラム 4.2 の実行結果で **default** 文が実行されてしまった理由は、負の数の入力により、剰余演算の結果として 0 でも 1 でもなく -1 という値が得られたためです。

解答プログラム 4.2　奇数／偶数判定プログラム（switch ～ case 版）

```
/************************************************
    p4_2.c
    偶数／奇数判定プログラム（switch ～ case 版）
    Tadaaki Shimizu    2006.11.10
************************************************/
#include <stdio.h>

main()
{
    int inputNumber;    // 入力される整数
    int remainder;      // 2 で割った余り

    // データの入力
    printf("Input Integer Number > ");
    scanf("%d", &inputNumber);

    // 偶数／奇数判定
    remainder = inputNumber % 2;
    switch(remainder) {
        case 0 : printf("%d は偶数です。\n", inputNumber);
                 break;
        case 1 : printf("%d は奇数です。\n", inputNumber);
                 break;
        default: printf(" 異常です。\n");
                 break;
    }
    exit(0);
}
```

[実行例]

```
% ./p4_2
Input Integer Number > 2
2 は偶数です。
% ./p4_2
Input Integer Number > 1
1 は奇数です。
% ./p4_2
Input Integer Number > -2
-2 は偶数です。
% ./p4_2
Input Integer Number > -1
異常です。
%
```

一方で、解答プログラム例 4.1 は負の数でも正しく動きます。これは、

```
if(inputNumber % 2 == 0)
```

という **if** 文で、先に偶数の場合 (いつでも剰余が **0** になる) を調べて、それ以外の場合を **else** 文で、奇数と判定しているためです。

```
if(inputNumber % 2 == 1)
```

という **if** 文で、先に奇数を判定するようにプログラムを書いていれば、プログラムは上手く動作しないでしょう。

もし、先に説明した負の数の割り算の事情を知った上で、プログラム 4.1 を書いたとしたら立派ですが、このような事情を知らずにプログラム例 4.1 を書いたとしたら、プログラムはたまたま運良く動いているにすぎません。

誰にも考え違いや、知らないことというのはあるものです。条件分岐には、自分の想定外の値が得られた場合を考慮した例外の分岐を付けることを習慣付けておくと、思わぬ落とし穴にはまらずに済みます。例えば、解答プログラム 4.1 の場合、条件分岐の部分を、次のように書いておくと良いでしょう。

```
if(inputNumber % 2 == 0)
    printf("%d は偶数です。\n", inputNumber);
else if(inputNumber % 2 == 1 || inputNumber % 2 == -1)
    printf("%d は奇数です。\n", inputNumber);
else
    printf("ERROR: 演算結果が異常です。\n");
```

演習問題 4.3

プログラム例4.7を改良して、剰余も求められるプログラムを作りなさい。

解答例　プログラムの作例を、解答プログラム 4.3 に示します。

この解答例での注目点は、2 点です。第一点は、**switch** ～ **case** 文を使ったプログラムに新たな分岐を組み込んで、機能を増やすことがいかに容易かという点です。解答プログラム 4.3 は、元になっているプログラム例 4.7 とほとんど変わりません。剰余演算を表すための定数を定義し、剰余演算を実行するための **case** 文を追加しただけです。**switch** ～ **case** 文は、その書式上、このような追加が非常に簡単に、読みやすさを失うことなく実行できます。その便利さを噛み締めて下さい。

第二点目は、些細な点ですが、**printf()** 関数の書式文字列によって、**%**(パーセント) を表示する方法です。解答プログラム 4.3 に示す通り、「**%%**」と 2 つ重ねることによって、一つの **%**(パーセント) を表示することができます。

解答プログラム 4.3　四則演算を行うプログラム

```
/************************************************
    p4_3.c
    四則演算を行うプログラム  ver1.4
        1から5の番号入力で演算を選択する（switch～case文）
    Tadaaki Shimizu    2006.11.10
************************************************/
#include <stdio.h>
#define ADD     1  // 足し算
#define SUB     2  // 引き算
#define MULT    3  // 掛け算
#define DIVI    4  // 割り算
#define MOD     5  // 剰余演算

main()
{
    int firstNum, secondNum;    //  入力される2整数
    int operation;              //  演算の種類
    int answer;                 //  演算の答え

    // データと演算種別の入力
    printf("Input Integer Number 1 > ");
    scanf("%d", &firstNum);
    printf("Input Integer Number 2 > ");
    scanf("%d", &secondNum);
    printf("Operation 1) +, 2) -, 3) *, 4) / 5) % >> ");
    scanf("%d", &operation);

    // 演算処理
    switch(operation) {
        case ADD:  answer = firstNum + secondNum;
                   printf("%d+%d = %d¥n",
                         firstNum, secondNum, answer);
                   break;
        case SUB:  answer = firstNum - secondNum;
                   printf("%d-%d = %d¥n",
                         firstNum, secondNum, answer);
                   break;
        case MULT: answer = firstNum * secondNum;
                   printf("%d*%d = %d¥n",
                         firstNum, secondNum, answer);
                   break;
        case DIVI: answer = firstNum / secondNum;
                   printf("%d/%d = %d¥n",
                         firstNum, secondNum, answer);
                   break;
        case MOD:  answer = firstNum % secondNum;
                   printf("%d%%%d = %d¥n",
                         firstNum, secondNum, answer);
                   break;
        default:   printf("Error: Operation (1-4)¥n");
    }
    exit(0);
}
```

演習問題 4.4

生年月日による次のような相性占いがある。この方法に従って、2人の生年月日を入力すると、相性を答えてくれるプログラムを作りなさい。

1）2 人のそれぞれの生年月日の各桁の数を全て足す。この 2 つの数の大きいほうから小さい方を引く。

2）1) で求めた数を 5 で割る。割り切れれば、相性は非常に良い。1 余れば、相性は良い。2 余れば、普通。3 余れば相性は悪い。4 余れば、相性は最悪。

解答例　プログラムの作成例を解答プログラム 4.4 に示します。

この演習問題は、本来は大変面倒な課題です。何故なら、生年月日という、形式があり制限がある入力を処理対象としているからです。実用のプログラムを作成するとしたら、入力された日付の妥当性のチェックなどで、大きな労力を必要とするでしょう。しかし、ここは演習問題ということで、簡単のためにチェックを行わず、ユーザーの入力を丸ごと信じることにしましょう。

入力は、年、月、日を別々に入力することにしました。この他に、年月日をまとめて、**19980112**(1998 年 1 月 12 日) のように、大きな整数として入力してもらう方法もあるでしょう。まだ、C 言語の勉強を始めたばかりなので、今までの知識でできることは限られますが、各自工夫してみて下さい。

生年月日の各桁の足し算は、商と余りの性質を使って計算しています。プログラムをよく読んでみて下さい。ほとんど同じ文が沢山並ぶ結果になってあまりカッコいいプログラムとは言えません。この後、繰り返し構文や、関数などを学ぶと、スマートに書き換えることができます。

解答プログラム 4.4　相性占いプログラム

```
/************************************************
    p4_4.c
    性格占いプログラム
    Tadaaki Shimizu    2006.11.10
************************************************/
#include <stdio.h>

main()
{
    int year, month, day;  //  成年月日
    int aPerson, bPerson;  //  成年月日の各桁合計
    int result;            //  相性の結果

    printf("1 人目 : 生まれた年 ( 西暦 ) > ");
    scanf("%d", &year);
    printf("1 人目 : 生まれた月 > ");
    scanf("%d", &month);
    printf("1 人目 : 生まれた日 > ");
    scanf("%d", &day);
```

```
    aPerson = year / 1000;
    year    = year % 1000;
    aPerson = aPerson + year / 100;
    year    = year % 100;
    aPerson = aPerson + year / 10 + year % 10;
    aPerson = aPerson + month / 10 + month % 10;
    aPerson = aPerson + day / 10 + day % 10;

    printf("2人目 : 生まれた年(西暦) > ");
    scanf("%d", &year);
    printf("2人目 : 生まれた月 > ");
    scanf("%d", &month);
    printf("2人目 : 生まれた日 > ");
    scanf("%d", &day);

    bPerson = year / 1000;
    year    = year % 1000;
    bPerson = bPerson + year / 100;
    year    = year % 100;
    bPerson = bPerson + year / 10 + year % 10;
    bPerson = bPerson + month / 10 + month % 10;
    bPerson = bPerson + day / 10 + day % 10;

    if(aPerson > bPerson) result = aPerson - bPerson;
    else                  result = bPerson - aPerson;
    result = result % 5;

    switch(result) {
        case 0: printf("相性は非常に良い。¥n");
                break;
        case 1: printf("相性は良い。¥n");
                break;
        case 2: printf("相性は普通。¥n");
                break;
        case 3: printf("相性は悪い。¥n");
                break;
        case 4: printf("相性は非常に悪い。¥n");
                break;
        default: printf("ERROR: 異常です。¥n");
    }
    exit(0);
}
```