文字データの取り扱い

演習問題 9.1　文法事項

1）コンピュータにおける、文字データの取り扱いについて説明しなさい。

解答例　コンピュータでは、文字に整数の番号を割り当てて (コード化して) 文字コードとして扱います。実際に用いられる文字コードとして、ASCII コード、EUC コード、JIS コード、SJIS コードなど、様々な規格が存在します。

2）C 言語の文字型は整数型の一種と考えられるが、その理由を述べなさい。

解答例　1) で答えたように、文字は文字コードとして扱われます。つまり、文字は小さな整数値で表されているのです。このため、文字型のデータの内容は、文字を表すコード番号であるという意味を除けば、単なる整数です。このため、C 言語の文字型は整数型の一種として考えることができます。

3）文字型のデータに対する四則演算は、実際にはどのような操作が行われるか答えなさい。

解答例　文字型のデータに対する四則演算は、文字コードをただの整数値と考えた場合の演算となります。例えば、文字型データどうしの足し算では、それぞれの文字のコード番号を整数として足し算したのと同じ結果が得られます。

4）標準文字処理関数などの標準関数を用いる利点を簡潔に述べなさい。

解答例　大文字と小文字の変換など、文字データに対する操作をプログラミングする場合、文字コードの性質を上手く利用しなければスマートなプログラムを書くことが難しくなります。しかし、一方で、文字コードの性質に依存したプログラムは、その依存性のために他のコンピュータへのプログラムの移植性が損なわれる恐れがあります。コンピュータのプログラムは、文字処理に関わらず、多くの機種依存性を持っており、プログラムの移植の妨げになります。

標準文字処理関数をはじめとする標準関数用いると、面倒な手続きを考えることなく、手軽に文字処理などの複雑な処理を行えます。また、標準関数の設計はよく考えられており、初心者が苦心して作ったプログラムより効率や信頼性が高いといえます。

さらに、標準関数は、C 言語を用いることができるコンピュータでは必ず利用できます。このため、標準関数を使った部分に関しては、他のコンピュータにプログラムを移植した際に、その動作を保証することができます。

以上のような、簡便さと移植性の高さから、可能な場合は、なるべく標準関数を用いてプログラムを設計することが望ましいといえます。

演習問題 9.2

プログラム例9.2を参考にして、aから zまでの文字の文字コード表を出力するプログラムを作りなさい。プログラムの実行例は、図9.8のようになります。(図9.8省略：本文を参照のこと)

解答例　解答例を解答プログラム 9.1 に示します。

このプログラムは、プログラム例 9.2 の処理を **for** 文によって繰り返しただけのもので非常に単純です。文字型 (**char** 型) の変数 **letter** を整数型の変数と同じようにカウンタ変数として使える点に注目して下さい。

文字コードを型変換した後に、特別に何か処理をするのでなければ、解答プログラム 9.2 のように、プログラムを簡略化することができます。**printf()** 関数への

解答プログラム 9.1　文字コード表を作るプログラム

```
/************************************************
    p9_1.c
    文字コード表を作る
    Tadaaki Shimizu    2006.11.17
************************************************/
#include <stdio.h>

main()
{
    char letter;    // 文字
    int  charCode;  // 文字コード

    for(letter = 'a'; letter <= 'z'; letter++) {
        charCode = (int)letter;
        printf("%c : %d\n", letter, charCode);
    }
    exit(0);
}
```

解答プログラム 9.2　文字コード表を作るプログラム 2

```
/************************************************
    p9_2.c
    文字コード表を作る 2
    Tadaaki Shimizu    2006.11.17
************************************************/
#include <stdio.h>

main()
{
    char letter;    // 文字

    for(letter = 'a'; letter <= 'z'; letter++)
        printf("%c : %d\n", letter, (int)letter);

    exit(0);
}
```

引数の引き渡しの際に型変換をすることで、プログラム中の変数を 1 つ減らすことができます。

解答プログラム 9.1 と解答プログラム 9.2 では、**a** から **z** までのアルファベットが文字コード上で順番に隙間無く並んでいることを仮定しています。従って、このような仮定が成り立たない文字コードを用いているコンピュータでは、このプログラムは正しく動作しません。この問題を解決する簡単な方法は、あるでしょうか？ あります！！

もし、文字が順番に並んでない可能性があるというならば、無理にでも順番に並べてしまえばよいでしょう。具体的には、配列を使って、配列の中に文字を **a** から順に **z** まで並べて格納してしまいます。このためには、次の第 10 章で学ぶ文字列を使うことができます。

（ここからの説明で理解できない点は、第 10 章を学んだ後に読んで下さい。）

改良したプログラムを、解答プログラム 9.3 に示します。

このプログラムでは、初期化付き宣言によって、**char** 型の配列 **letter[]** にアルファベットの列を文字列として格納します。配列 **letter[]** について、**for** 文により、**0** 番目の要素から、**'\0'**（ヌル文字：文字列の終端記号）の前まで、順に文字と文字コード番号を表示します。

解答プログラム 9.3 は、文字コードが前述の仮定を満たしていない場合でも、正しく動作します。しかも、解答プログラム 9.1 と解答プログラム 9.2 と比べて、たいして複雑になっているわけでもありません。シンプルでかつ移植性も高いという意味では、非常に良いプログラムだということができます。

解答プログラム 9.3　**文字コード表を作るプログラム 3**

```
/************************************************
    p9_3.c
    文字コード表を作る 3
    Tadaaki Shimizu    2006.11.17
************************************************/
#include <stdio.h>

main()
{
    char letter[] = "abcdefghijklmnopqrstuvwxyz"; // 文字
    int  i;                                       // カウンタ変数

    for(i = 0 ; letter[i] != '\0'; i++)
        printf("%c : %d\n", letter[i], (int)letter[i]);

    exit(0);
}
```

演習問題 9.3

2つの文字を入力し、その間のアルファベットを順に出力するプログラムを作りなさい。例えば、eと iが入力されると、e,f,g,h,iの5文字が出力されます。ただし、2つの文字の入力の順番は問わないように、プログラムを作りなさい。

解答例 この問題は単純なのですが、やってみると面倒くさいことが山積みです。文字の入力だけでも、考えなければならないことが沢山あります。演習問題では、プログラムの詳細については指定していませんので、問題にチャレンジする皆さんが自分自身で考える必要があります。以下に挙げる点は、あくまでも例ですので、参考程度に思って下さい。

1) アルファベット以外の文字が入力されたらどうする？

この解答例では、エラーメッセージを表示して終了することにしました。

2) 大文字と小文字の扱いはどうする？

この解答例では、全て小文字に直して扱うことにしました。

解答例を、解答プログラム 9.4 に示します。

このプログラムでは、変数に **scanf()** 関数を使って 1 文字入力しています。その後、**isalpha()** 関数によって、入力された文字がアルファベットかどうか調べ、アルファベットでなければ、プログラムを終了しています。さらに、**tolower()** 関数を使って小文字にします。

上記の文字入力を変数 **startLetter** と **endLetter** について行った後、変数 **startLetter** の方に順番の若いアルファベットが入るように入れ替えをします。ここでは、「順番が若いアルファベットほど値が小さい文字コードである」という仮定をしています。

文字の表示では、**for** 文を用いて、カウンタ変数 **i** により、**startLetter** に格納された文字から、**endLetter** に格納された文字の 1 つ前までを表示します。さらに、**for** 文を抜けた後、もう一度 **printf()** 関数で **i** の内容を **"%c"** によって表示します。このとき、**i** の値は **endLetter** の値と一致しているので、きちんと最後の文字が表示されます。このように、ちょっと面倒な表示方法を取ったのは、文字と文字の間に「,(カンマ)」を表示し、最後の文字の後で改行するという表示形式にしたかったからです。

このプログラムでは、「文字の入力」の部分では、標準関数を用いてプログラムの機種依存性を取り除いていますが、「文字の入れ替え」と「文字の表示」の部分では、文字コードの性質 (若い順に連続にコード化されている) を直接使っています。この依存性を取り除くことは、演習問題 9.3 の解答例で示したような方法で可能ですが、プログラムが複雑にかつ遅くなります。しかも、少なくとも英文字のアルファベットについては、ここで仮定した性質を持たない文字コードは考えられないので、よほどのことがない限り大丈夫でしょう。

連続したアルファベットを表示するプログラム

```
/************************************************
    p9_4.c
    連続したアルファベットを出力
    Tadaaki Shimizu    2006.11.20
************************************************/
#include <stdio.h>
#include <ctype.h>

main()
{
    char startLetter, endLetter;  // 開始／終了文字
    char tmp;                     // 入れ替え用のテンポラリ変数
    int  i;                       // カウンタ変数

    // 文字の入力
    printf(" 1文字入力 : ");
    scanf("\n");   // 1文字入力のためのおまじない
    scanf("%c", &startLetter);
    if(!isalpha((unsigned char)startLetter)) {
        printf("ERROR : 入力はアルファベットではありません。\n");
        exit(0);
    }
    startLetter = (char)tolower((unsigned char)startLetter);

    // 文字の入力
    printf(" 1文字入力 : ");
    scanf("\n");   // 1文字入力のためのおまじない
    scanf("%c", &endLetter);
    if(!isalpha((unsigned char)endLetter)) {
        printf("ERROR : 入力はアルファベットではありません。\n");
        exit(0);
    }
    endLetter = (char)tolower((unsigned char)endLetter);

    // 文字の入れ替え
    if(startLetter > endLetter) {
        tmp         = startLetter;
        startLetter = endLetter;
        endLetter   = tmp;
    }

    // 文字の表示
    for(i = startLetter; i < endLetter; i++)
        printf("%c,", i);
    printf("%c\n", i);

    exit(0);
}
```

[ちょっと難しい話題]

解答プログラム 9.4 では、**isalpha()** 関数などを用いるときに、

```
isalpha((unsigned char)endLetter)
```

としています。この中で気になるのは、**(unsigned char)** というキャスト演算子ではないでしょうか。**char** 型が整数型の一種であるということは、本文中で述べました。多くのコンパイラでは、**char** 型は、**-128** から **127** までを表すことができる符号付きの整数型として扱われます。一方、文字データはコード番号であり正の数です。**char** 型の変数は文字コードの入れ物としては、符号無しの整数と同じように使われています。

char 型のデータを文字のデータとしてだけ使っている場合は、**char** 型が符号付きか符号無しかということはあまり問題になりません。しかし、**char** 型を **int** 型に型変換する場合は大問題を引き起こします。

0 から **127** までの文字コードは、**char** 型から **int** 型に型変換する場合、その値を保ったまま変換されます。しかし、**128** 以上の文字コードは、問題を起こします。**128** 以上の文字コードは、**char** 型を符号ありと考えると負の数になってしまいます。このため、符号付きの **char** 型の変数に格納された **128** 以上の文字コードは、**int** 型に型変換すると、負の数になってしまいます。

この事態を防ぐには、**char** 型を一度強制的に符号無しにしてしまえば良いのです。このためには、**(unsigned char)** キャスト演算子を用います。上記の **isalpah()** 関数の呼び出しでは、変数 **endLetter** に格納されたデータを、キャスト演算子で一旦、符号無し **char** 型 (**unsigned char** 型) に強制的に型変換し、その後、暗黙のうちに **int** 型に型変換してから、渡しているのです。これによって、**128** 以上の文字コードも正しく扱うことができます。

本書の例題や演習問題では、コメントや **printf()** 関数による表示 (書式文字列) を除いては、文字として英数字だけを想定しています。それでも、面倒なことは幾つもあります。ましてや、日本語など英語以外の文字を扱うとなると大変です。文字や文字列の扱いは、プログラムを書く者にとって、思った以上に厄介なことなのです。

演習問題 9.4

1桁の16進数を入力すると、対応する10進数を表示するプログラムを作りなさい。16進数と10進数の対応を、表9.5に示します。(表9.5省略：本文を参照)

解答例 解答例を解答プログラム 9.5 に示します。16 進数を、単なる英数字と考えて、**char** 型のデータとして扱いました。1 文字入力の扱いについては、演習問題 9.3 で詳しく述べたので省略します。

プログラムは色々と書きようがあるでしょうが、ここでは、文字データとして入力された 16 進数を、**int** 型の整数に直して、その結果を表示するようにしました。「数値に変換」という部分と「結果の出力」の部分を完全に分けてプログラムを作ったので、多少の無駄もありますが、わかりやすくはなっているのではないでしょうか？このプログラムでは、数字や英文字の文字コードが順番に連続して並んでいること

解答プログラム 9.5　16 進数と 10 進数の変換

```
/************************************************
    p9_5.c
    16進数
    Tadaaki Shimizu    2006.11.20
************************************************/
#include <stdio.h>
#include <ctype.h>
#define ERROR_INPUT -1  // エラーコード ( 入力エラー )

main()
{
    char hexadecimal;    // 16 進数
    int  decimal;        // 10 進数

    // 16 進数 : 文字として入力
    printf(" 16 進数 1 桁入力 : ");
    scanf("\n");    // 1 文字入力のためのおまじない
    scanf("%c", &hexadecimal);
    hexadecimal = (char)tolower((unsigned char)hexadecimal);

    // 数値に変換
    if(isdigit((unsigned char)hexadecimal)) // 0 - 9 の場合
        decimal = (int)(hexadecimal - '0');
    else if(hexadecimal >= 'a' && hexadecimal <= 'f')
        decimal = (int)(hexadecimal - 'a') + 10;
    else
        decimal = ERROR_INPUT;

    // 結果の出力
    if(decimal != ERROR_INPUT)
        printf("%c : %d\n", hexadecimal, decimal);
    else
        printf("ERROR: 入力が 16 進数ではありません。\n");

    exit(0);
}
```

を仮定しています。そんなことを全く仮定せずに、どんな文字コードでも動くようにする方法もあります。プログラム例 9.6 を見て下さい。泥臭い力技ですが、扱う数字が 16 種類程度と少ない場合には、よい方法です。

解答プログラム 9.6　**16 進数と 10 進数の変換**

```
/************************************************
    p9_6.c
    16 進数
    Tadaaki Shimizu    2006.11.20
************************************************/
#include <stdio.h>
#include <ctype.h>
#define ERROR_INPUT -1  // エラーコード ( 入力エラー )

main()
{
    char hexadecimal;   // 16 進数
    int  decimal;       // 10 進数

    // 16 進数 : 文字として入力
    printf(" 16 進数 1 桁入力 : ");
    scanf("\n");   // 1 文字入力のためのおまじない
    scanf("%c", &hexadecimal);

    // 数値に変換
    switch(hexadecimal) {
        case '0': decimal = 0;
                  break;
        case '1': decimal = 1;
                  break;
        case '2': decimal = 2;
                  break;
        case '3': decimal = 3;
                  break;
        case '4': decimal = 4;
                  break;
        case '5': decimal = 5;
                  break;
        case '6': decimal = 6;
                  break;
        case '7': decimal = 7;
                  break;
        case '8': decimal = 8;
                  break;
        case '9': decimal = 9;
                  break;
        case 'a':
        case 'A': decimal = 10;
                  break;
        case 'b':
        case 'B': decimal = 11;
                  break;
```

```
        case 'c':
        case 'C': decimal = 12;
                  break;
        case 'd':
        case 'D': decimal = 13;
                  break;
        case 'e':
        case 'E': decimal = 14;
                  break;
        case 'f':
        case 'F': decimal = 15;
                  break;
        default : decimal = ERROR_INPUT;
                  break;
    }

    // 結果の出力
    if(decimal != ERROR_INPUT)
        printf("%c : %d\n", hexadecimal, decimal);
    else
        printf("ERROR: 入力が 16 進数ではありません。\n");

    exit(0);
}
```