ファイルの入出力

演習問題 12.1 文法事項

1) プログラム中でファイルへの入出力を行うために必要な基本操作の概要を簡潔に答えなさい。

解答例 C 言語でファイル操作を行うには、まずファイルを開き (オープン)、入出力を行い、その後ファイルを閉じる (クローズ) する必要があります。

ファイルのオープンには、`fopen()` 関数を用い、入出力には、`fputc()` 関数や `fgetc()` 関数をはじめとする多彩な入出力関数を用いることができます。ファイルのクローズには、`fclose()` 関数を用います。

2) プログラム中でファイルへの入出力を行うために、ファイルを指定するためにどのような方法を用いますか。

解答例 プログラム中でファイルの指定 (識別) を行うために、ファイル・ポインタを用います。ファイル・ポインタは、ファイルの各種情報を格納しているファイル構造体へのポインタ (`FILE *`) であり、`fopen()` 関数によってファイルをオープンする際に、返り値として取得することができます。

3) ファイルに様々なデータを格納するための、2 つの方法について、その得失を含めて簡潔に説明しなさい。

解答例 ファイルにデータを格納する方法として、テキスト形式で格納する方法とバイナリ形式で格納する方法があります。

テキスト形式は、データを一旦文字に変換する形式です。例えば整数データであれば、`"12"` のように数字の並びに変換します。この方法の最大の利点は、人間がデータをそのまま読み取れることです。エディタ等を用いて、データを編集することもできます。欠点は、データを文字に変換する際に数値の精度が失われる可能性があることと、データ量 (ファイルの大きさ) が多くなってしまうことです。

一方、バイナリ形式では、データを、コンピュータが内部を扱う形式で、そのままファイルに出力します。例えば、整数データの場合、計算機内部で扱っている 2 進数符号のままファイルに書き出します。バイナリ形式の利点は、データの変換を伴わないため、データの精度を失う恐れが無いことと、データ量 (ファイルの大きさ) が小さくて済むことです。欠点は、ファイルに格納されたデータを人間が見ても、全く意味が判らなず、扱いづらいことです。

テキスト形式とバイナリ形式は、プログラムの用途とその運用方法などによって、適切に使い分けなければなりません。

演習問題 12.2

プログラム例 12.1 を参考にして、テキスト・ファイル中の行数を数えるプログラムを作りなさい。テキスト・ファイル中で、行末には、改行文字 ('\n')があることを思い出してください。

解答例　プログラム例を解答プログラム 12.1 に示します。

このプログラムは、本文のプログラム例 12.1 とほとんど同じです。異なる部分は、文字が読み込まれる度に変数 **count** をインクリメントするのではなく、読み込んだ文字が **'\n'** の場合だけインクリメントするようにしてあることだけです。これにより、ファイルの内容の行数を数えることができます。

解答プログラム 12.1　行数を数えるプログラム

```
/************************************************
    p12_1.c
    ファイルに含まれる行数を数えるプログラム
    Tadaaki Shimizu    2007.01.11
************************************************/
#include <stdio.h>

main()
{
    char fileName[100];  // ファイル名
    FILE *inputFp;       // 入力ファイル・ポインタ
    int  count;          // 行数を数える
    int  letter;         // 読み込んだ文字

    printf("Input file name: ");     // ファイル名入力
    scanf("%s", fileName);
    inputFp = fopen(fileName, "r"); // ファイルのオープン
    if(inputFp == NULL) {           // ファイル・オープン失敗の場合
        printf("%s が開けません \n", fileName);
        exit(0);
    }

    // 行数を数える
    count = 0;
    for(;;) {
        letter = fgetc(inputFp);
        if(letter == '\n')      count++;
        else if(letter == EOF) break;
    }
    fclose(inputFp);                 // ファイルのクローズ

    printf("%s の中身は %d 行です。\n", fileName, count);
    exit(0);
}
```

演習問題 12.3

プログラム例12.2を参考にして、2つのテキスト・ファイルの内容をつないで、別のファイルに書き込むプログラムを作りなさい。

解答例 プログラム例を解答プログラム 12.2 に示します。
本文中のプログラム例 12.2 が理解出来ていれば、このプログラムは簡単でしょう。
2 つの入力ファイルに対して同じ操作を繰り返すだけです。

解答プログラム 12.2　ファイルの接続

```
/************************************************
    p12_2.c
    ファイルの接続を行うプログラム
    Tadaaki Shimizu    2007.01.11
************************************************/
#include <stdio.h>
#define FOREVER 1

main()
{
    char firstFileName[100];    // １つ目の入力ファイル名
    char secondFileName[100];   // ２つ目の入力ファイル名
    char outputFileName[100];   // 出力ファイル名
    FILE *firstFp;              // １つ目の入力ファイル・ポインタ
    FILE *secondFp;             // ２つ目入力ファイル・ポインタ
    FILE *outputFp;             // 出力ファイル・ポインタ
    int  letter;                // 読み込んだ文字

    printf("1つ目のコピー元ファイル名： ");  // 入力ファイル名入力
    scanf("%s", firstFileName);
    firstFp = fopen(firstFileName, "r"); // 入力ファイルのオープン
    if(firstFp == NULL) {   // オープン失敗の場合
        printf("%s が開けません \n", firstFileName);
        exit(0);
    }
    printf("2つ目のコピー元ファイル名： ");  // 入力ファイル名入力
    scanf("%s", secondFileName);
    secondFp = fopen(secondFileName, "r"); // 入力ファイルのオープン
    if(secondFp == NULL) {   // オープン失敗の場合
        printf("%s が開けません \n", secondFileName);
        fclose(firstFp);     // すでに開いている入力ファイルを閉じる
        exit(0);
    }

    printf("コピー先ファイル名： ");  // 出力ファイル名入力
    scanf("%s", outputFileName);
    outputFp = fopen(outputFileName, "w"); //出力ファイルのオープン
    if(outputFp == NULL) {   // オープン失敗の場合 */
        printf("%s が開けません \n", outputFileName);
        fclose(firstFp);     // すでに開いている入力ファイルを閉じる
        fclose(secondFp);
        exit(0);
    }
```

```
    while(FOREVER) {   // １つ目のファイルから文字をコピーする
        letter = fgetc(firstFp);
        if(letter == EOF) break;
        fputc(letter, outputFp);
    }
    fclose(firstFp);

    while(FOREVER) {   // ２つ目のファイルから文字をコピーする
        letter = fgetc(secondFp);
        if(letter == EOF) break;
        fputc(letter, outputFp);
    }
    fclose(secondFp);
    fclose(outputFp);

    exit(0);
}
```

演習問題 12.4

キーボードから入力された整数を、1行に1つずつテキストとしてファイルに書き込むプログラムを作りなさい。ただし、データの入力前に、ファイル名とデータの個数を入力しておくようにしなさい。

解答例　プログラム例を解答プログラム 12.3 に示します。

このプログラムでは、**scanf()** 関数によってキーボードから読み込まれた整数データを変数 **data** に格納し、これを **fprintf()** 関数によってテキスト形式でファイルに書き出しています。

プログラムを簡単にするために、先にデータ数を入力するような仕様にしましたが、プログラムの設計としてはあまり良いことではありません。プログラムの利用者が予めデータの数を数えておかないとプログラムを使えないからです。

より良いプログラムにするためには、このような利用者の負担を減らす設計にする必要があります。具体的には、利用者が何らかの終了操作を行った場合に、入力を終了するように設計します。例えば、利用者が数字以外の文字を入力したら、入力を終了するなどです。これによって、利用者がデータ数を数える必要が無くなります。

また、このプログラムでは、整数データに対して特別な演算は行っていません。キーボードから入力してファイルに出力するだけです。このとき、キーボードから入力された文字データ (数字の並び) は、**scanf()** 関数によって int 型のデータに変換され変数 **data** に格納します。その後、変数 **data** に格納された **int** 型データを、**fprintf()** 関数によって文字データに変換されてファイルに出力します。

このように説明すると、無駄が見えてきましたね？　データは文字の並びから整数へ、整数から文字の並びへと、変換が繰り返されます。しかし、ただファイルに書き出すだけなら、一旦 **int** 型データに変換する必要はありません。入力された文字の並びをそのままファイルに書き出しても同じ結果となります。

とは言え、ここは練習問題ですから、解答プログラム 12.3 を正解としておきましょう。

解答プログラム 12.3　整数データのファイル

```
/************************************************
    p12_3.c
    整数のデータファイルを作る
    Tadaaki Shimizu    2007.01.11
************************************************/
#include <stdio.h>

main()
{
    int  data;                  // 入力された整数データ
    int  numOfData;             // データ数
    char outputFileName[100];   // 出力ファイル名
    FILE *outputFp;             // 出力ファイル・ポインタ
    int  i;                     // カウンタ変数

    printf("コピー先ファイル名 : ");  // 出力ファイル名入力
    scanf("%s", outputFileName);
    outputFp = fopen(outputFileName, "w"); //出力ファイルのオープン
    if(outputFp == NULL) {  // オープン失敗の場合 */
        printf("%s が開けません \n", outputFileName);
        exit(0);
    }
    printf("入力データ数 : "); // データ数入力
    scanf("%d", &numOfData);
    if(numOfData <= 0) {
        printf("データ数は 0 より大きくして下さい。\n");
        exit(0);
    }

    for(i = 0; i < numOfData; i++) {
        printf("データ入力 [%d 個目 ] : ", i + 1);
        scanf("%d", &data);
        fprintf(outputFp, "%d\n", data);
    }
    fclose(outputFp);

    exit(0);
}
```