ファイル

これまでにデータの入力方法として、キーボードからの入力を用いてきた。構造体を習った際に実感してもらえたと思うが、入力データ量が多いときにはその作業は大変なものとなり、入力するデータを間違えた場合には最初からやり直しになる。そこで今回はこれらの問題を解決するため、あらかじめ入力データをテキストエディタなどで編集し、ファイルとして保存したものを入力データとして用いる方法を習っていく。さらにプログラムで作成したデータをファイルに出力する方法も併せて習っていく。

C 言語におけるファイルでの入出力には、通常の文書として見ることができるテキストファイルと、2 進数のデータの羅列として記録されるバイナリファイルの 2 種類が用意されている。テキストファイルは一見してデータの内容を把握できるが、ファイルのサイズがバイナリファイルに比べて大きくなる。また、バイナリファイルではファイルのサイズが比較的小さくなるが、データの内容はバイナリエディタなどで確認することになり、その場合でも内容を正確に把握することは難しい。ここでは取り扱いが簡単なテキストファイルの取り扱いについて学んでいく。

ファイルの操作

ファイルとはWindowsやlinuxで採用されているデータの管理システムとして扱うことができるひとまとまりのデータの単位であり、日常生活におけるノートのような役割を持つ。このファイルはハードディスクなどの記憶装置に保存され、読み・書き・実行に利用される。C 言語のプログラムにおいて、このファイルの読み書きを行う場合には以下の操作に従う。

（1）ファイルを開く
（2）ファイルの読み/書きを行う
（3）ファイルを閉じる

ファイルの操作に関する関数はいくつか用意されているが、ここではfopen（開く）、fclose（閉じる）、fprintf（書く）、fscanf（読む）の４つについて学んでいく。

ファイルの読み込み

それでは具体的な処理として、ファイルからデータを読み込み、プログラム内でそのデータを利用するプログラム例を示していく。

プログラム例１

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
        char fn[] = "indata0.dat";
        int val, num, sum;
        FILE *fp;

        if((fp = fopen(fn, "r")) == NULL)
```

```
        {
                printf("ファイル%sが見つかりません。¥n", fn);
                exit( EXIT_FAILURE );
        }

        num = 0; sum = 0;
        while(1)
        {
                fscanf(fp, "%d", &val);
                if(val == 0) break;
                num++;
                sum += val;
        }

        printf("読み込み回数:%d 合計:%d¥n", num, sum);

        fclose(fp);

        return EXIT_SUCCESS;
}
```

実行結果例

```
読み込み回数:5 合計:1725
```

このプログラムは入力データを記述したファイル「indata0.dat」からデータを読み込み、読み込んだデータ数、読み込んだデータの合計を求めて表示するプログラムである。このプログラムを実行する際には以下のようにして、「indata0.dat」というファイルをテキストエディタで編集して保存したものをプログラム例１の実行ファイルと同じフォルダに用意してから実行する必要がある。今回の場合、ファイルの最後には 0 を必ずつけることで、読み込むデータが終了したことを把握出来るようにしている。

入力データファイル「indata0.dat」

```
123
234
345
456
567
0
```

ファイルを開く処理は以下の書式に従って行う。

書式

```
fp = fopen(fn, "r");
```

fn はファイル名を保存してある文字列の先頭アドレスである。"r"は read の頭文字で読み込みモードを設定している。fp は FILE 型ポインタで宣言した変数名である。この FILE 型ポインタは「ストリームポインタ」とも呼ばれる。この処理を行った場合、ファイルの読み込み処理が失敗する場合がある。それは読み込みたいファイルが存在しなかったり、読み込みを制限する設定が行われたファイルだった

り、ハードウェアやシステムの問題だったりと様々な理由があるが、その場合には変数 fp には「NULL」が格納される。この状態で以降のプログラムを行うことは危険であるため、読み込みが成功したかどうかのエラーチェックと供に用いるのが通常である。エラーチェックを行った読み込み処理はプログラム中でも示しているが、以下の通りになる。

```
if((fp = fopen(fn, "r")) == NULL)
{
        printf("ファイル%sが見つかりません。¥n", fn);
        exit( EXIT_FAILURE );
}
```

これは以下と同じ処理である。

```
fp = fopen(fn, "r");

if(fp == NULL)
{
        printf("ファイル%sが見つかりません。¥n", fn);
        exit( EXIT_FAILURE );
}
```

この使用方法で示した exit()関数はプログラムを終了させる処理を行う。この関数は stdlib.h に用意されているため、インクルードが必要となる。また、()内の EXIT_FAILURE は 1 に置き換えることも可能であり、その場合にはプログラムが異常終了したことを表す。環境によってはプログラムを実行したターミナル上に「プログラムが異常終了しました」という内容の表示を行うため、問題の特定に役立つ。また、プログラムの最後に今までは return 0;と書いてきたが今回、return EXIT_SUCCESS;と用いた。これを使うと処理の意味が分かりやすい。

次にファイルの読み込みについて示す。ファイルの読み込みには fscanf を用いた。基本的な使い方はキーボードからデータを読み込む際に用いた scanf と同様である。違いは関数の引数の 1 番目に FILE 型ポインタである変数 fp を書く必要があることである。

書式
fscanf(fp, "%d", &val);

最後にファイルを閉じる処理を示す。以下のプログラムの実行によって、ファイルが閉じられる。

書式
fclose(fp);

このファイルを閉じる処理はファイルを読み込む際には大きな問題にはならない。しかし、後述するファイルの書き込み処理で行わなかった場合、データを正しくファイルに保存することが保証されない。

それでは次のプログラムを確認しよう。「indata0.dat」ファイルはファイルの最後に 0 を必ずつ

けるというルールでファイルを作成したが、これでは使い勝手が悪い。実際にはファイルの最後かどうかの判断は別の処理を利用する。

プログラム例２

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
        char fn[] = "indata.dat";
        int val, num, sum;
        FILE *fp;

        if((fp = fopen(fn, "r")) == NULL)
        {
                printf("ファイル%sが見つかりません。¥n", fn);
                exit( EXIT_FAILURE );
        }

        num = 0; sum = 0;
        while(fscanf(fp, "%d", &val) != EOF)
        {
                num++;
                sum += val;
        }

        printf("読み込み回数:%d 合計:%d¥n", num, sum);

        fclose(fp);

        return EXIT_SUCCESS;
}
```

実行結果例

```
読み込み回数:5 合計:1725
```

このプログラムでは以下のように最後の 0 を削除したファイル「indata.dat」ファイルを用意すること。

入力データファイル「indata.dat」

```
123
234
345
456
567
```

プログラム例１との違いは while 文の部分であり、以下のようになっている。

```
while(fscanf(fp, "%d", &val) != EOF)
{
        num++;
```

```
        sum += val;
}
```

条件式に用いているが、データの読み込みに用いる fscanf()関数はファイルの読み込み時にチェックを行い、ファイルの最後だった場合には返却値として「EOF」を返す。これは「End Of File」という意味がある。

ファイルの書き込み

それでは、次にファイルの書き込みを行うプログラム例を示す。これはプログラム例２についてディスプレイに出力した結果と同じものをファイルに保存する処理を追加したプログラムである。

プログラム例３

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
        char ifn[] = "indata.dat";
        char ofn[] = "outdata.dat";
        int val, num, sum;
        FILE *ifp, *ofp;

        if((ifp = fopen(ifn, "r")) == NULL)
        {
                printf("ファイル%sが見つかりません。¥n", ifn);
                exit( EXIT_FAILURE );
        }

        num = 0; sum = 0;
        while(fscanf(ifp, "%d", &val) != EOF)
        {
                num++;
                sum += val;
        }

        fclose(ifp);

        if((ofp = fopen(ofn, "w")) == NULL)
        {
                printf("ファイル%sが見つかりません。¥n", ofn);
                exit( EXIT_FAILURE );
        }

        printf("読み込み回数:%d 合計:%d¥n", num, sum);
        fprintf(ofp, "読み込み回数:%d 合計:%d¥n", num, sum);

        fclose(ofp);

        return EXIT_SUCCESS;
}
```

実行結果例

読み込み回数:5 合計:1725

まずはファイルを開く処理について確認する。書式は読み込み処理を行う場合に似ているが、書き込みモードで行うため、"r"だった部分が"w"に置き換わっている。"w"は write の頭文字である。

書式
ofp = fopen(ofn, "w");

プログラム中ではエラーチェックと供に記述したが、使い方は読み込みモードの場合と同じである。次に実際に書き込みを行っている処理について見ていく。書き込みは fprintf()関数を用いているが、これは printf()関数の使い方と似ている。違う点はやはり FILE 型ポインタ変数 ofp を第一引数に持つことである。

書式
fprintf(ofp, "読み込み回数:%d 合計:%d¥n", num, sum);

最後にファイルを閉じるが必要であるため、以下の処理を記述する。

書式
fclose(ofp);

読み込み処理のところでも触れたが、ファイルの書き込み処理における fclose()関数は重要である。なぜなら、ファイルへの書き出しはある程度のデータがバッファに貯まった段階で行われる。バッファはデータを一時的に蓄えておく場所である。よってファイルを行わない場合にはバッファに蓄えられたデータを書き出すタイミングを失うことになるため、データが正しくファイルに書き出されない状況が発生する。ファイルを扱う処理では、オープン・読み/書き・クローズの 3 つの処理は必ずセットで行うことが原則である。

ファイルへの追記

すでにデータが記述されたファイルにデータを追加したい場合がある。その場合にはファイルを追記モードとして開くことになる。追記モードの利用は"a"を使って以下のように記述する。"a"は addition の頭文字である。

書式
ofp = fopen(ofn, "a");

この処理以降は fprintf()関数を使って、通常の書き込み処理と同様に行えばよい。例えば、プログラム例３において、上記の書式に変更した場合には実行するたびにファイルにデータが追加されるため、以下のようなファイルが作成されることになる。

実行結果例
読み込み回数:5 合計:1725

```
読み込み回数:5 合計:1725
読み込み回数:5 合計:1725
        :
```

課題５

Kadai5-1　プログラム例 1～2 を作成し、実行結果を確認する。その上で、入力ファイル名をキーボードから入力するプログラムに変更しなさい。提出はプログラム例 2 をキーボードから入力できるようにしたものだけでよい。

Kadai5-2　プログラム例 3 を作成し、実行結果を確認する。また、書き込みモード"w"を追記モード"a"に変更したものを実行しなさい。その上で、入力ファイル名と出力ファイル名をキーボードから入力するプログラムに変更しなさい。提出はキーボードから入力できるようにしたものだけでよい。

Kadai5-3　プログラムの実行ファイルを実行した回数をカウントするプログラムを作成しなさい。これは実行した回数をファイルに記録することで実現する。例えば、data.dat ファイルを用意し、中身を 0 としておく。プログラム実行時にはこのファイルのデータを読み込み、+1 してから、ファイルにその結果を書き出せばよい。この場合にはファイル名をプログラム内で指定している方が良い。

Kadai5-4　ファイル処理を用いたオリジナルのプログラムを作成しなさい。

締め切りについては後で示す。