ファイル入出力

山本昌志*

2004 年 11 月 09 日

1　前回の復習と今週の内容

1.1　前回の復習

配列を学習するまで、諸君が知っていたデータ構造は、単純型と呼ばれるもので、

```
int a;
double x;
```

のように宣言する。これは、

- a という名前が付いた整数を格納する入れ物を用意する。
- x という名前が付いた倍精度実数を格納する入れ物を用意する。

というように解釈する。このデータ構造では、変数名、たとえば a や w を指定することで、データにアクセスできる。

先週は、単純型に比べ、より多くのデータを扱える配列というデータ構造について、学習した。それは、

```
int b[10000];
double y[10000];
```

のように宣言する。配列名と自然数により目的のデータにアクセスできる。この配列を上手に使うことにより、大量のデータを取り扱うことができる。このようなことを学習した。

1.2　今週の学習内容

前回までの学習で、配列を使うことにより、大きなデータを扱えるようになったが、依然としてデータの入力は、キーボードを用いていた。また、データの表示もディスプレイのみであった。これでは、実際には大きなデータを扱うことは難しい。そこで、本日は、データをハードディスク上で読み書きする方法を学習する。

*国立秋田工業高等専門学校　電気情報工学科

2　ファイルの種類

2.1　ファイルと C 言語プログラム

1 つの単位として取り扱えるデータの集合をファイルと言い、その集合には名前 (ファイル名) が付けられている。通常は、ハードディスクのような外部記憶装置上に保存されている。ファイルはオープンすることによって、ストリーム[1]と関連付けられる。

C プログラムがファイルをオープンすると、ファイルと C プログラムは、ストリームによって結ばれる。これを、図??に示す。C プログラムとファイルを結びつけるものがストリームというものである。このストリームと言う概念は、他のプログラミング言語でも出てくるので覚えておいた方が良い。

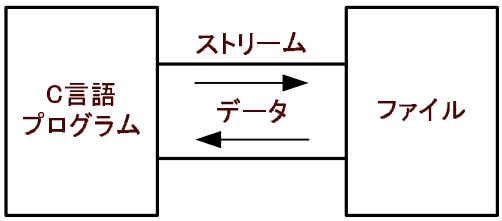

図 1: プログラムとストリーム、ファイルの関係

ここでは、ほとんどの場合、ファイルはハードディスクを示すが、実際にはディスプレイやキーボード、メモリーなど様々なデバイスである。

2.2　バイナリーファイルとテキストファイル

ストリームに流れるデータは、バイナリーデータとテキストデータがある。バイナリーデータ[2]の場合、コンピューターで取り扱っているデータそのものである。一方、テキストデータの場合、データは ASCII 文字に変換される。これでは、何のことか分からないと思うので、データ (整数の 1234 と文字列”akita”) をテキストとバイナリーで保管されたファイルを、図 2 と 3 に示す。テキストデータの方は、通常のテキストエディターで内容を見ることができるが、バイナリーデータの方は意味不明の文字が書かれる。バイナリーデータを見るためには、図 5 のようにバイナリーエディターで見るしかない。それでも、書かれている内容を理解することは難しい[3]。

ようするに、テキストデータの方が簡単である。この授業ではテキストデータしか取り扱わないことにする。

[1] stream:流れ
[2] binary:二進数の
[3] $(1234)_{10} = (4d2)_{16}$ で、リトルエンディアンで記録している

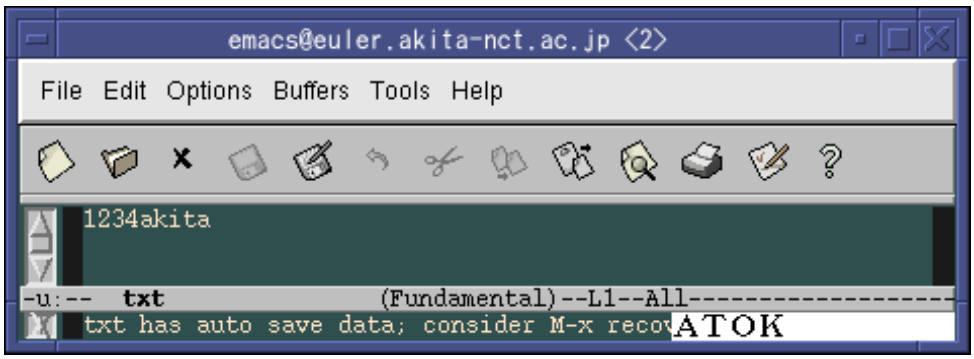

図 2: テキストデータ。整数の 1234 と文字列”akita”。

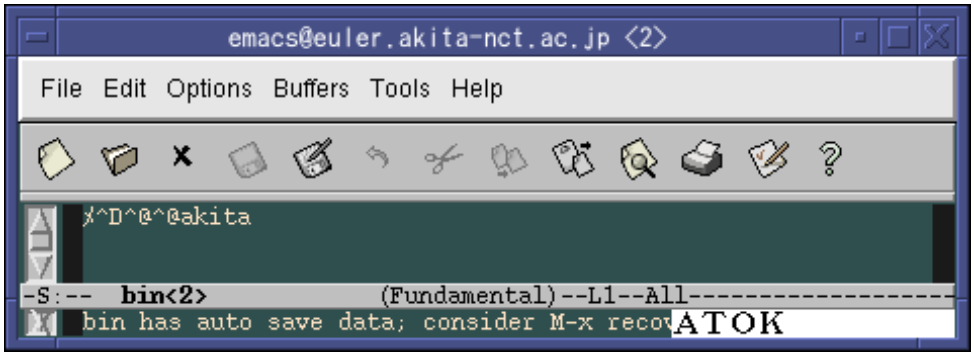

図 3: バイナリーデータ。整数の 1234 と文字列”akita”。

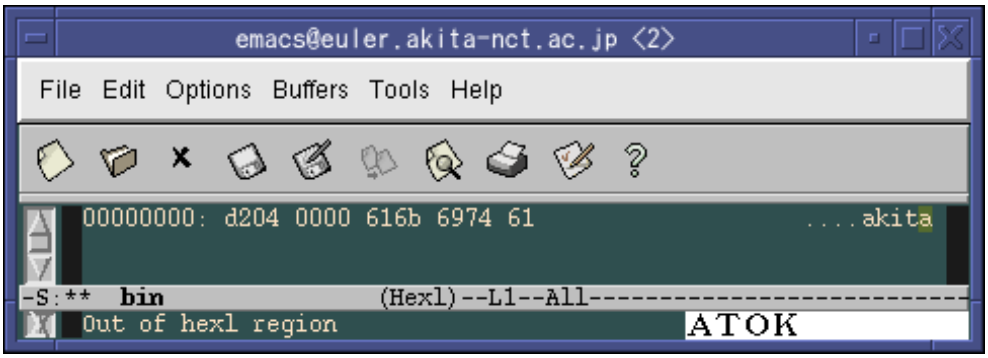

図 4: バイナリーデータをバイナリーモードで表示。

2.3　シーケンシャルアクセスとランダムアクセス

ハードディスク上に書かれたデータは、長い巻物に文字が書かれていると思えばよい。巻物であるから、ページの概念はないが、行はある。また、1 行に書かれる文字数の制限も無いと考えてよい。

この巻物に大量の文字が書かれているとしよう。この文字を、巻物の始めから、そして左から順番に読むのをシーケンシャルアクセスと言う。一方、あっちこっちに飛んで、いろいろな場所をつまみ食いのように読むのをランダムアクセスと言う。これらの違いを理解しておくことが大事である。

この授業では、取り扱いが簡単なシーケンシャルアクセスのみを学習する。ランダムアクセスについては、興味のある者は、自分で学習せよ。

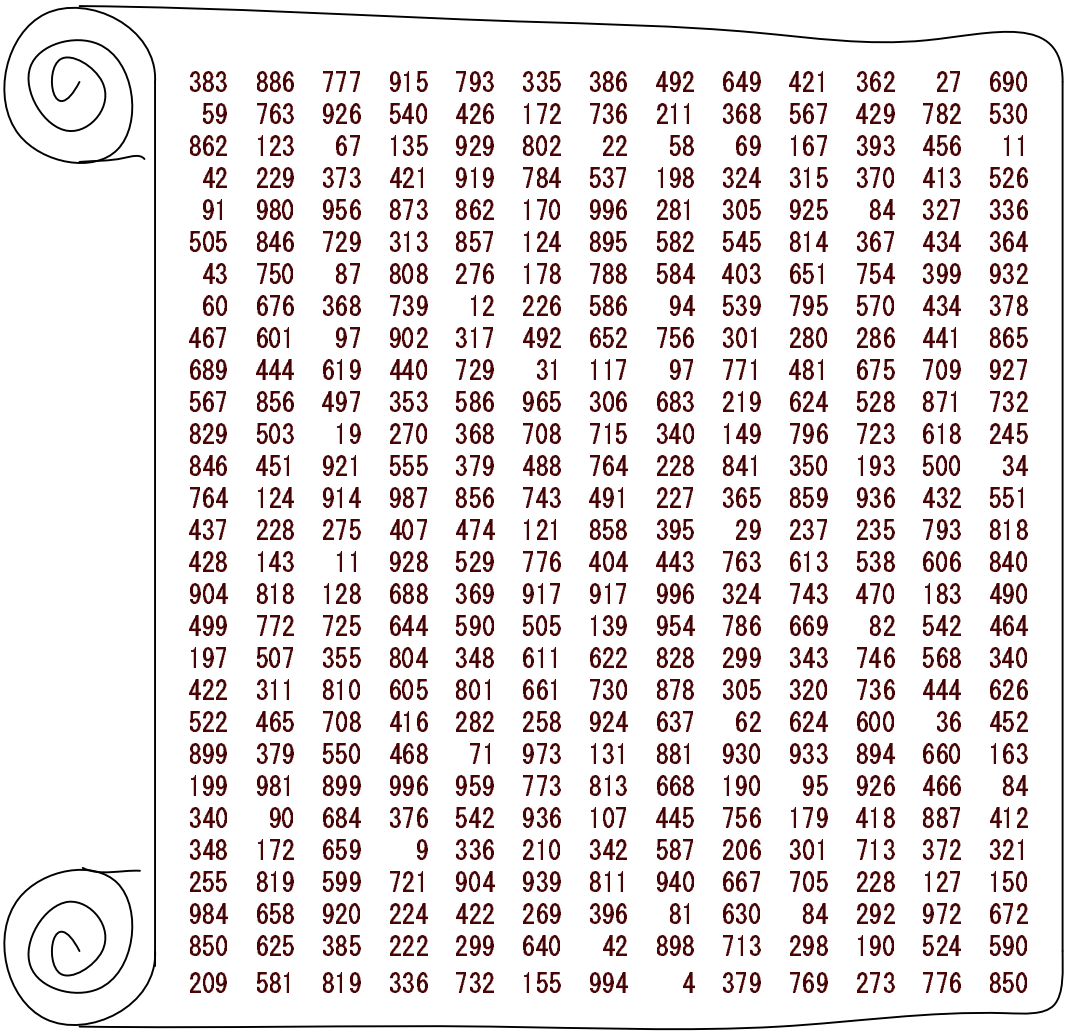

図 5: ファイルのイメージ。

3　ファイル入出力の例

3.1　ファイル出力

細かい文法の話をする前に、実際のファイル出力の例を示す。以下のプログラムがファイル出力の例である。このプログラムを実行すると、”output.txt”というテキストファイルができる。このテキストファイルは、図 6 のようになっている。

このプログラムの中で、ファイル処理に関する部分は、次の通りである。

- 「FILE *fp」でファイルポインター fp を宣言している。ファイルポインターとはファイルをの情報を入れておく変数だと思ってほしい[4]。
- 「fp = fopen("output.txt","w")」で”output.txt”と言うファイルを、書き込みモード ("w") で開いている。そのファイルの情報は、fp に代入される。
- 「fprintf(fp,"%3d\t", 10*i+j)」で、ファイルに書き込んでいる。ファイルポインターを示す引数以外は、printf 文と同じである。
- 「fclose(fp)」でファイルを閉じている。

実際のファイルの書き込みは、printf 文とほとんど同じなので、簡単である。ただ、前後にファイルのオープンとクローズがあるだけである。

```
#include <stdio.h>

int main(void)
{
  int i, j;
  FILE *fp;

  fp = fopen("output.txt","w");

  for(i=0;i<=99;i++){
    for(j=0;j<=9;j++){
      fprintf(fp,"%3d\t", 10*i+j);
    }
    fprintf(fp,"\n");
  }

  fclose(fp);

  return 0;
}
```

[4]実際は、ポインターなので変数を入れる場所を示すものである。ポインターについては学習指定なので、この様に書くと覚える。

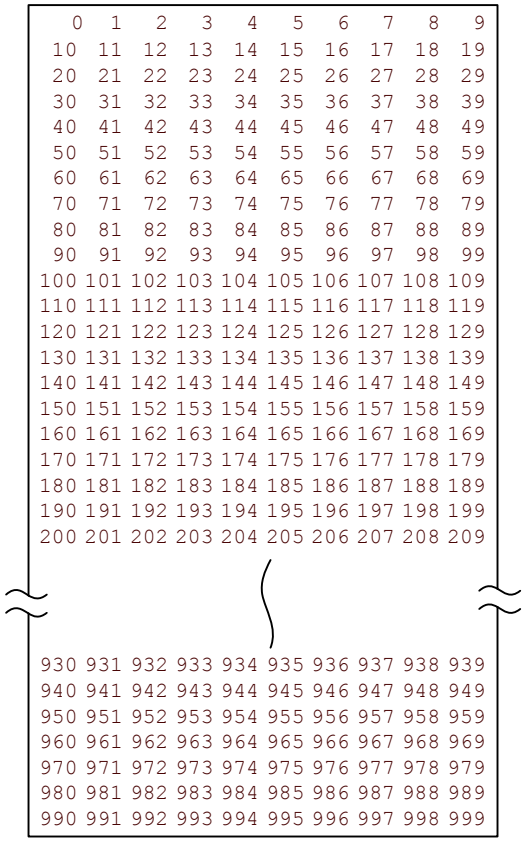

図 6: 作成されたテキストファイル。

3.2 ファイル入力

ファイル入力は、次のように書く。このプログラムを実行すると、先ほどファイル出力で作成されたファイルのデータを読み込み、配列 data に数値が代入される。

このプログラムの中で、ファイル処理に関する部分は、次の通りである。

- 「FILE *fp」でファイルポインター fp を宣言している。
- 「fp = fopen("output.txt","r")」で”output.txt”と言うファイルを、読み込みモード ("r") で開いている。そのファイルの情報は、fp に代入される。
- 「fscanf(fp,"%d", &data[i][j])」で、配列に代入している。ファイルポインターを示す引数以外は、scanf 文と同じである。
- 「fclose(fp)」でファイルを閉じている。

実際のファイルの読み込みは、scanf 文とほとんど同じなので、ファイル出力同様に簡単である。

```
#include <stdio.h>

int main(void)
{
  int i, j;
  int data[100][10];
  FILE *fp;

  fp = fopen("output.txt","r");

  for(i=0;i<=99;i++){
    for(j=0;j<=9;j++){
      fscanf(fp,"%d",&data[i][j]);
    }
  }

  fclose(fp);

  return 0;
}
```

4 C 言語でのファイル入出力

4.1 ファイル入出力の流れ

どのような言語でもファイル処理の流れは似ている。まず初めにファイルを開いて (open)、その後処理を行い、最後に閉じる (close) のである。その流れを図 7 に示す。

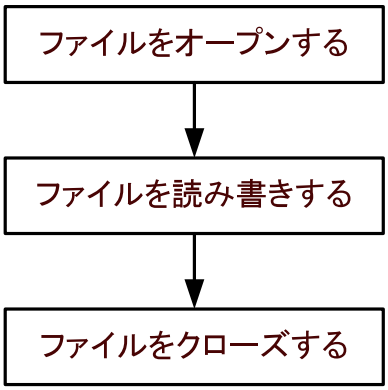

図 7: ファイル処理の流れ。

4.2 ファイルポインター

ストリームを制御するために必要な情報が記述されている。これは変数の宣言と同じで、次のように書く。とりあえず、ファイルを使うときに必要な変数と思ってほしい。

```
FILE *ファイルポインター名;
```

注意すべきことは、

- 変数の型名は、すべて大文字で「FILE」と書く。
- ファイルポインター名の前に、アスタリスク「*」をつける。

である。

4.3 オープンとクローズ

ファイルを使うときには、最初にそれをオープンする必要がある。

```
ファイルポインター名 = fopen("ファイル名","モード");
```

ファイル名は、絶対パスや相対パスで記述することができる。また、ただ単にファイル名のみがかかれた場合、その実行ファイルがあるディレクトリーとなる。モードは、バイナリーとテキストがある。テキストの場合について、表 1 に示す。いろいろなモードがあるが、とりあえずは、読み込み (r) と書き込み (w) を理解しておけばよい。

表 1: テキストモードのオープン

モード	処理	ファイルがないとき	ファイルがあるとき
r	読み込み (read)	NULL を返す	正常処理
w	書き込み (write)	新規作成	前のファイルは破棄
a	追加書き込み (append)	新規作成	前の内容の後に追加
r+	書き込み (更新)	NULL を返す	正常処理
w+	読み込み (更新)	新規作成	前の内容は破棄
a+	追加読み書き	新規作成	前の内容の後に追加

使い終わったならば、クローズしなくてはならない。

```
fclose(ファイルポインター名);
```

4.4 ファイル出力

ファイル出力は、printf 文とほとんど同じである。

```
fprintf(ファイルポインター, 出力書式, 引数並び);
```

4.5 ファイル入力

ファイル入力は、scanf 文とほとんど同じである。

```
fscanf(ファイルポインター, 入力書式, &引数並び);
```

4.6 特別なファイル

UNIX には、ファイルポインターの指定やオープン、クローズ処理なしで使えるファイルがある。それを、表 2 に示す。「fscanf」や「fprintf」のファイルポインター名を書くところに、これらを書くと、キーボード入力やディスプレイ出力になる。上手に使えば、便利である。

表 2: 特殊なファイル

ファイル	ファイルポインター	デバイス
標準入力	stdin	キーボード
標準出力	stdout	ディスプレイ
標準エラー出力	stderr	ディスプレイ

5　予習

5.1　プログラム作成

以下のプログラムを次回の授業で作成する。次回の授業まで、プログラムの内容を考えてくること。今回は課題としないが、次回の授業で課題を与える。

- 11 月の毎日の 1 時間毎の気温のデータファイルがある。表 3 のようになっている。
  - 各行には、その日の 1 時間毎の 24 個のデータがある。0 時〜23 時までである。
  - 行数は 30 行で、11 月 1 日から 11 月 30 日を表している。
- 日毎の最高気温と最低気温、平均気温をディスプレイに書き出す。
- 11 月の最高気温と最低気温、平均気温をディスプレイに書き出す。

表 3: 11 月の気温

8.3	7.9	7.5	7.2	$\cdots$	9.8
9.3	9.2	9.1	9.0	$\cdots$	6.3
6.2	5.8	5.3	4.9	$\cdots$	12.0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
4.3	3.9	3.3	2.8	$\cdots$	3.8