課題C言語のプログラムについての説明

山本昌志*

2005年9月22日

1　前期末試験に向けて

これまで学習してきたC言語の全てが、前期末試験の範囲である。90%以上は、夏休みの課題のプログラムから出題する。前々回の授業で解答のプログラムも渡しているので、十分学習して試験に臨むこと。

2　質問事項の説明

前回のプログラムの実習の時間での質問事項をまとめておく。

2.1　乱数の発生

問題の 2.3 配列 [練習 2] で使われている乱数の部分が分かりません。

乱数とは、バラバラな数列のことを言う。とくに、自然数がめちゃくちゃに現れるようなものを自然乱数という。0以上無限大までの全ての自然数を用いた自然乱数が考えられるが、実際上は最大の自然数を決め、その範囲で考えることが多い。我々が使用しているシステムのC言語の場合、0〜2147483647の範囲[1]の乱数を発生させることができる。

C言語のプログラムでは、`rand()` 関数を使って、乱数を発生させる。例えば、次のようにすると、`rand()` 関数が呼び出される度に、それが乱数を返し、配列 `a[i]` に格納される。

```
for(i=0; i<ndata; i++){
  a[i]=rand();
}
```

コンピューターは正確に言われたとおり (プログラムのとおり) に計算を行うことは、諸君もよく知っているはずである。そのため、コンピューターはめちゃくちゃな順序で数が並んでいる乱数を発生させることは苦手である。先ほどの `rand()` 関数は、ある初期値[2]を使って、計算に

*独立行政法人　秋田工業高等専門学校　電気情報工学科

[1] $0 \sim 2^3 1\text{-}1$ の範囲である。

[2] 正確には seed(種) と言うらしい。

より乱数を決めている。同じ初期値を使って、rand() 関数を呼び出すと、同じ数列が発生するこのになる。これでは、乱数とは言い難いので、初期値を毎回変更するのがよい。

初期値も値が毎回異なる整数を決める必要があるが、現在の暦時刻を返す time() 関数を用いるのが一般的である。初期値の設定は、srand() 関数に引数 (符号無し整数) を渡すことにより可能である。次のようにすれば、毎回異なる初期値を決めることができる。

```
srand((unsigned int)time(NULL));
```

ただし、1 秒以内であれば、time は同じ値となり、同じ初期値となり、同じ乱数となることに注意が必要である。(unsigned int) は、キャストと呼ばれる強制型変換で、引き続く値の型を変換している。time() 関数の引数は暦時刻で、暦時刻がポインターで格納される。暦時刻を格納する必要がないときには、NULL と空ポインターを指定する。

以上から、乱数を発生させるためには、rand() と srand()、time() 関数が必要であることが分かった。これらの関数を使うためには、関数の宣言が書かれているヘッダーファイルが必要である。rand() と srand() には stdlib.h が、time() には time.h が必要となる。以上をまとめると、配列 a[i] に 1024 個の乱数を発生させるためには、次のように書く。

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

int main(void){
  int a[1024], i;

  srand((unsigned int)time(NULL));

  for(i=0; i<1024; i++){
    a[i]=rand();
  }

  return 0;
}
```

2.2 平方根

問題の 2.6 関数 [練習 1] で使う平方根 ($\sqrt{\ }$) の計算方法が分かりません。

数学で使う平方根は、関数を表すことは十分知っていると思う。$\sqrt{x}$ と書けば、x の 1/2 乗を計算する。したがって、C 言語では通常の関数のように平方根を書くことになるが、$\sqrt{\ }$ という記号を使うことは出来ない。C 言語の関数名は、アルファベットの小文字 (a〜z) か大文字 (A〜Z)、あるいはアンダースコア (_) を使って表現しなくてはならないからである。このようなことから、数学の $\sqrt{\ }$ の代わりに、sqrt() 関数[3]を使う。$y = \sqrt{x}$ を書きたければ、次のようにするのである。

[3]sqrt と言うのは、square root (平方根) の略である。

```
y=sqrt(x);
```

もちろん、これは数学関数であるので、ヘッダーファイルの math.h にその宣言が書かれている。そのため、プログラムの最初の方に

```
#include <math.h>
```

と記述し、コンパイル時には、gcc -lm -o hoge fuga.c のように、-lm というオプションを付ける。

2.3 マクロ定義#define

問題の 2.2 関数 [練習 5] で使われている#define TEST 100000 の意味が分かりません。

#define はプリプロセッサーと呼ばれるもので、2.7 構造体 [練習 1] や 2.8 ポインター [練習 2] 等で使っている。プリプロセッサーというのは、コンパイルに先立って、ソースファイルのテキストを編集するものである。#include もそのひとつである。

ここでは、#define を文字列置換として使っている。例えば、練習問題の解答プログラムで使っている

```
#define TEST 100000
```

のような場合の動作は次の通りである。このプリプロセッサーが書かれている後の文では、TEST という文字列は全て、100000 に置き換わる。コンパイルに先立って、この置き換えの動作が実施されるので、文字列 TEST は 100000 と同じ扱いになる。

この文字列置換は便利な機能で、一回で多くの文字列を置き換えてくれる。例えば、ここでは 100000 までの素数を求めるのであるが、200000 までと問題が変更された場合、プログラムの変更は 1 箇所で済む。#define TEST 200000 とすればよいのである。これは、ソースファイル全てにわたって実施されるので、関数内でのみ有効というわけではない。

通常、重要な数がプログラム中にそのまま記述されるのは望ましくない。その数が変更になると、複数の文を直す必要が生じ、バグの元である。そこで、#define を使って、重要な数は 1 箇所で記述するのがセオリーである。

また、#define の直後に書かれる文字は、マクロ名と呼ばれ大文字で書くことが慣例となっている。C 言語の関数や変数名は小文字で書かれることが慣例となっており、それがむやみに置き換わらないようにするための配慮である。

2.4 データの終わりを示す EOF

問題の 2.4 ファイル入出力 [練習 2] で使われている EOF の意味が分かりません。

この練習問題では、ファイルのデータの読み込みと出力の部分は、次のように書かれている。

```
while(fscanf(fp, "%d%lf%lf%lf", &theta, &s, &c, &t) != EOF){
  printf("%d\t%f\t%f\t%f\n", theta, s, c, t);      /* 表示 */
}
```

このこの部分の動作は、次のようになっている。

1. fscanf() 関数で書式に従いファイルからデータを読む。
2. fscanf() 関数の戻り値が EOF と異なるならば、
   - printf() 関数により、書式に従いデータを出力する
   - 再度、2 に戻る (fscanf() 関数を実行する)。

   を実行する。もし、戻り値が EOF ならば、while のループを抜ける。

fscanf() 関数の戻り値が EOF になったならば、読み込みと出力の動作の繰り返し文から抜ける構造となっている。fscanf() 関数の戻り値は、入力項目数である。このプログラムの場合は、データがある場合は 4 が返される。

EOF という戻り値は、fscanf() 関数で読み込むデータが無いときの値である。全てデータを読み終わって、ファイルの終わりにきたときに EOF が返されるのである。データ数が不明で、ファイルの終わりまでそれを読み込みたい場合、EOF を使うのが常套手段である。

EOF は、End Of File からきており、その値は-1 となっていることが多い。

2.5 キャスト

問題の 2.2 制御文 [練習 5] で使われている (int) の意味が分かりません。

これは、キャスト (強制型変換) と呼ばれる演算子である。この練習問題のプログラム中では、次のように使われている。

```
test_max=(int)sqrt(TEST);
```

ここで、変数 test は整数型であり、TEST は #define により 100000 である。したがって、$\sqrt{100000}$ を計算し、それを整数化 (切り捨て) した後、整数変数 test_max に代入することになる。sqrt(TEST) の戻り値は倍精度実数であるので、強制型変換演算子 (int) により、整数化している。

このように式[4]の値の型を変えたい場合にキャストという強制型変換を使う。これは、次のように書けばよい。

(型名) 式

[4]変数単独、あるいは関数の戻り値も式と見なす。

2.6 文字読み込み

問題の 2.5 文字処理 [練習 1] で使われている fgets の動作が分かりません。

fgets() 関数は、ファイルから文字数を指定して、文字列を読み込む関数である。その書式は、以下の通りである。

```
fgets(配列名, 配列の大きさ, ファイル)
```

戻り値は、読み込みに成功したら入力文字の先頭アドレスを格納しているポインター (配列名) を、失敗したら NULL を返す。

練習のプログラムでは、次のように使われている。

```
fgets(moji, 32, stdin);
```

moji[32] で宣言されている配列に、ファイル stdin からデータを読み込むようになっている。ファイル stdin は特別なファイルで、標準入力 (STanDard IN) で、通常はキーボードを表す。

scanf() 関数を使わない理由は、空白 (スペース) を含めた文字列を配列に格納したいからである。scanf() 関数の場合、空白は文字列の区切りとして取り扱われるので、それを配列に入れることは出来ない。

キーボードから空白を含めた文字列を入力するだけならば、gets() という関数もあるが、これは使わないのが慣例である。この関数は、配列で確保された以上のデータを入力するとプログラムが暴走する可能性がある。