C言語プログラミングの基礎訓練

2018 年 2 月 22 日版

西井 淳

目次

1 プログラム作成上の注意点

1) 以下でプログラムを作る時には，Makefile を作成し，**make コマンドでコンパイル**できるようにする。
2) 言語は C++ でもよい。(この場合ファイル入出力の命令は，練習問題中の指定とは異なる C++ のストリーム入出力の命令を用いてよい)
3) その他，別冊の「C 言語プログラミングの掟」をよく読んでから以下の課題に取り組む

こと

2 準備運動 (この節のプレゼン発表会します)

2.1 コンパイル・リンク

[問] C 言語等のプログラムを書いたとき、それを実行できる形式にするには**コンパイル**と**リンク**が必要である。この「コンパイル」と「リンク」とはどのような作業をすることか説明しなさい。説明には「プリプロセッサ」,「ヘッダファイル」,「ソースプログラム」,「オブジェクトファイル」,「実行形式」というキーワードを使うこと。

2.2 変数とポインタ

以下の各プログラムを実行したとき、各変数のための記憶領域がメモリ空間でどのように確保されて値の受け渡しが行われ，結果はいかに表示されるかを説明しなさい。

問 1

```
1  #include <stdio.h>
2  int z=1;
3
4  void func(int *x)
5  {
6    static int y=8;
7    extern int z;
8    printf("%D,%D,%D\N",(*x)++,y++,z++);
9  }
10 int main(void)
11 {
12   int x=2,y=5,z=4;
13   func(&y);
14   printf("%D,%D,%D\N",x++,y++,z++);
15   func(&z);
16   printf("%D,%D,%D\N",x++,y++,z++);
17   return(0);
18 }
```

問 2

```
1  #include <stdio.h>
2
3  void func(int n1, int *np2)
4  {
5    n1=*np2;
```

```
6     *np2=8;
7     np2=&n1;
8   }
9
10  int main(void)
11  {
12    int n1=0, n2=5;
13    int *np1, *np2, *tmp;
14
15    np1=&n2;
16    n2++;
17    tmp=&n1;
18    (*tmp)++;
19    np2=tmp;
20    (*tmp)++;
21
22    printf("N1=%D, N2=%D, *NP1=%D, *NP2=%D\N", n1, n2, *np1, *np2);
23
24    func(*np1,np2);
25    printf("*NP1=%D, *NP2=%D\N", *np1, *np2);
26  }
```

3 基本

3.1 定数の定義

定数はプリプロセッサ (付録参照) を使って

```
1  #define MAX 10
```

と定義する方法と、const を使って以下のように定義する方法がある。

```
1  const int MAX=10;
```

#defineは**プログラムのコンパイル前に単なる文字列置換として実行**され、**const**を用いて宣言した変数は**プログラムの実行時にメモリ領域が確保**される。 このような特徴のため，それぞれ長所と短所がある。

1) 上述した**#define**と **const**の仕様を厳密に満たすコンパイラにより以下のプログラムをコンパイルすると、値が正常に表示されない。その理由を述べ、修正方法を 2 通り (**#define**を使う方法と **const**を使う方法) を述べなさい。

```
1  #include <stdio.h>
2  #define MIN -2
3  int main(void)
```

```
4 {
5     printf("%D",3/MIN);
6 }
```

2) 下記のプログラムをコンパイルするとコンパイルエラーが出たり、コンパイルは出来ても実行時に配列 aの値がおかしくなる等の不具合が出ることがある。 配列の大きさはコンパイル時に決まっていないといけないことに注意し，なぜこのような不具合が起きるか説明しなさい。また，**#define**を使ってプログラムを修正しなさい。

```
1 #include <stdio.h>
2 const int MAX=3;
3 int main(void)
4 {
5     int a[MAX],i;
6     for(i=0;i<MAX;i++) a[i]=0;
7     ...
8 }
```

3.2 double と float の演算

1) **double** x=0.0に 0.1 を 10 回足した値は 1.0 にならない．いったいどの程度その値は違うのだろう？ またその理由はなにか？
2) 前問で **double** のかわりに **float** を用いた場合について，同様のことを議論せよ．
3) 以下のプログラムを作ったところ暴走してしまった．修正案を考えなさい。

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5    double x=0.0;
6    while(1) {
7      x+=0.1;
8    if(x==1.0) break;
9    }
10
11   return(0);
12 }
```

4) x=0.1を 0.1 倍してから 10.0 倍するという演算を何度か繰り返す場合に付いて，xが double の場合と float の場合でどのくらい演算時間の差があるかを調べよ。プログラムの実行時間は time コマンドで計測できる ($ time <command name>)。

コメント) 結果は処理系によって違うが，**double**と**float**の演算にかかる時間は同じになる処理

系も多い。また、**float**は精度が非常に悪いため、メモリ量が少ないとき以外には**float**はほとんど使われない。

3.3 main 関数への引数の処理

1) プログラムに与えた引数の数と，引数を表示するプログラム showargを作りなさい。実行結果の例は以下の通り。

```
1 $ ./showarg a b c
2 4
3 ./showarg a b c
```

2) 引数として与えた 2 つの実数の和を出力するプログラムを作りなさい。($ ./sum 1 2 と実行すれば"3" と表示されるプログラム)。引数が 2 個与えられなかったときには，以下のようなメッセージを出して終了すること。

```
1 $ ./sum 1
2 Usage: sum <num1> <num2>
```

注）main 関数への引数は全て文字列として受け取られる。文字列を **double**に変換するには関数atof() を用いる。

3.4 動的なメモリ確保

1) 引数で整数 n が与えられたとき，calloc を用いて配列 **int** a[n]を確保し，全ての $i < n$ について a[i]=i を代入した後，a[] の内容を表示するコマンド arrayをつくりなさい。なお，この節でのプログラムでは以下に気を付けること。
   a) メモリ確保に失敗したときには，エラー出力を行って終了すること。
   b) 必ず確保したメモリーは必ずプログラム終了時までに開放すること。
2) 引数で整数 n が与えられたとき，２次元配列 **int** a[n][n]を確保し，その配列を単位行列として、配列 aの内容を表示するコマンド array2をつくりなさい。

4 ファイル入出力と分割コンパイル

1) 以下の関数群をつくり，mylib.cという名前で保存しなさい。
   a) 引数で指定したファイル名の関数を読込みモードで開き，そのファイルへのファイルポインタを返す関数 fRopen。ファイルを開くのに 失敗したときには、“Failed to

open ○○”(○○には引数で与えたファイル名が入る) と**標準エラー出力**に出力してプログラムを終了。

```
1   FILE* fRopen(char* fname)
```

b) 引数で指定したファイル名の関数を書込みモードで開き，そのファイルへのファイルポインタを返す関数 fWopen。ファイルを開くのに失敗したときには、“Failed to open ○○” と**標準エラー出力**に出力してプログラムを終了。

```
1   FILE* fWopen(char* fname)
```

2) mylib.cの中の関数のプロトタイプ宣言 (関数原型宣言) のみを記載したファイル mylib.hをつくりなさい。

3) make cleanを実行したら，末尾に~ がついた名前のファイルと拡張子が.oであるファイルが削除されるように Makefile を作りなさい。

5 乱数とデータ処理

1) 引数で与えた回数だけ 0 以上 1 以下の一様乱数を発生し、結果を各行に表示するプログラム genrandをつくりなさい。また，100 回乱数を発生させた結果をリダイレクト (UNIX 関連のドキュメント参照) を用いて保存しなさい。

2) 数値データをファイルから読み込み，以下を行うプログラム getdistをつくりなさい。 ただし，以下の仕様を満たすこと。

b) –hオプションもしくは、オプションが不適切なときには，使い方 (Usage) を以下のように表示して終了する。

```
1  $ ./getdist -h
2    Usage: getdist [option] <file>
3    option:
4    -h) Show this message
5    -n) with line number
6    ...(適宜追加)...
```

c) –a オプションでデータの平均，標準偏差，最小値，最大値を表示

d) –g オプションでデータのヒストグラムを出力 (設定した刻み幅で, 度数頻度を’*’ を使って表示する)。例えば設定した刻みを 0.1 にした場合には

```
1      0-0.1:    **
2      0.1-0.2: ****
3      0.2-0.3: ***
4      ...
```

といった表示を行えるようにする。刻み幅の数は**#define**で用いて定義することにより，可変にする。

e) C 言語でのオプション処理は通常の文字列処理関数を使って行っても，gcc の標準ライブラリ getoptを使っても良い。getoptの使い方は各自調べること。

f) プログラムのはじめのほうで，どのような引数があるかをチェックし，引数が不適切な場合には，ただちに Usage を表示して終了する。

g) 既に作った mylib.cにあるファイルを開く関数を利用し、分割コンパイルを行うこと。

h) make getdistを実行すればコマンドgetdistを出力できるように Makefile を作りなさい。

3) getdist .cを改造して，マクロ変数 FILE(付録参照) を定義しているときにはファイルresult .datに結果を出力するようにし，定義してないときにはこれまで通り標準出力に結果を表示するようにしなさい。

ヒント：FILE を定義した場合には出力ファイルポインタを指定したファイルに、定義していないときには標準出力 (stdin) に設定する。**#ifdef**の分岐は一カ所のみですむはず。

5.1 パイプ (発展課題)

1) 関数 fRopenをパイプ (付録参照) に対応できるようにした関数 fRPopenを作り，mylib.c, mylib.hに追加しなさい。

2) パイプを用いてデータファイルを受け取れるように，前問で作成したgetdistを改良したgetdist2を作りなさい。その動作は以下を試して確認しなさい。

```
1 $ genrand 100 | getdist2
2 $ genrand 100 | getdist2 -g
3 $ getdist2 <file name>
4 $ getdist2 -g <file name>
```

6 微分方程式

1) 微分方程式 $\dot{x} = x$ について以下を行いなさい。

a) 微分方程式の解を（解析的に）求めなさい。また，その解の挙動のグラフを書きなさい。

b) 微分方程式の解 $(x(t))$ を求めるプログラムを作成し，それをもとに自然定数 e を求めなさい。

2) 質量 m の物体を地表から角度 θ 上方に初速度 v_0 で投げた時の軌道を計算したい。

a) 物体の運動方程式を書きなさい。

b) 数値計算により，物体の軌道を求めなさい。数値計算のプログラムは以下のようにすること。
   - 各パラメータ値は静的変数として，ヘッダ部で定義する。
   - 離散計算の時間刻みも静的変数としなさい。
   - 計算終了は物体が地面に落下時にすること。

c) 質量 5 kg の物体が 45° の角度で時速 100km で投げた時，どれだけ遠くに届くかを求めなさい。ただし空気抵抗は無視できるとし，重力加速度の大きさは $g = 9.8\ \mathrm{m/s^2}$ とする。
   - 離散計算で得られる値と，理論値を比べ，どの程度の精度で計算できるかを調べること。
   - 離散計算の時間刻みを変えた時に，その精度がどう変化するかも確認しなさい。

7 このドキュメントの著作権について