これまでの復習 (前期中間試験に向けて)

山本昌志*

2004 年 6 月 3 日

これまでの、学習をまとめます。中間試験には、このプリントに書いてある内容を理解して臨むこと。中間テストの実施要領は、以下の通りである。

- 教科書は持ち込み可とする。
- 配布したプリントはは持込不可とする。教科書にプリントのコピーを貼り付けたものは、カンニングと見なす。ただし、教科書への書き込みは可とする。

1 重要な UNIX コマンド

- UNIX のファイル構造は、ツリー (木) 構造です。
  - 今、自分が居るディレクトリーを、カレントディレクトリーと言います。カレントディレクトリーのパス (位置) を調べるコマンドは、「`pwd`」です。
  - パスを表す場合、デイレクトリーの区切りには「`/`」(スラッシュ) を使います。
  - カレントディレクトリーを明示したい場合は、1 つのピリオド「`.`」で表します。
  - カレントディレクトリーの 1 つ上のディレクトリーを親デレクトリーと言います。親ディレクトリーへ移動するコマンドは、「`cd ..`」です。2 つのピリオド「`..`」が、親ディレクトリーを表します。
  - カレントディレクトリーの直ぐ下のディレクトリーを子ディレクトリー、あるいはサブディレクトリーと言います。例えば、子ディレクトリー hogehoge に移動するコマンドは、「`cd hogehoge`」です。
  - ユーザー各個人が使用 (読み、書き、実行) を許されている最上位のディレクトリーをホームディレクトリーと言います。移動するコマンドは、「`cd`」です。
  - 2 通りのパスの表し方があります。例えば、私のホームディレクトリー (`/home/tea/yamamoto`) にサブディレクトリー (`sub_1`)、そのサブディレクトリー (`sub_11`) は、次のように表すことができます。相対パスはカレントディレクトリーからの相対位置を表し、ここではホームディレクトリーとする。

    絶対パス　`/home/tea/yamamoto/sub_1/sub_11`

    相対パス　`sub_1/sub_11`　あるいは　`./sub_1/sub_11`

*国立秋田工業高等専門学校　電気工学科

- カレントディレクトリーにあるファイルやサブディレクトリーの名前を調べるコマンドは、「`ls`」です。
- サブディレクトリ hogehoge の追加と削除のコマンドは、以下の通り。
- ファイルを削除するコマンドは、「`rm hogehoge`」です。これで、hogehoge と言うファイルが削除されます。
- 空っぽのサブディレクトリー (`hogehoge`) を削除するコマンドは、「`rmdir hogehoge`」です。サブディレクトリーに中身が有る場合は、「`rm -rf hogehoge`」とします。サブディレクトリーに含まれるもの全て削除されます。
- ファイルやディレクトリーをコピーするコマンドは、「`cp hogehoge hugahuga`」です。これで、hogehoge というファイルあるいはディレクトリーの hugahuga という名のコピーが作成されます。
- ファイルやディレクトリーを移動するコマンドは、「`mv hogehoge ../hugahuga`」です。親ディレクトリー (`..`) にサブディレクトリー hugahuga が無い場合、hogehoge が親ディレクトリーに移動して、名前を hugahuga に変更されます。もし、親ディレクトリー (`..`) にサブディレクトリー hugahuga がある場合、hugahuga のサブディレクトリーとして、hogehoge と言う名前で移動します。
- 以前使用したコマンドを呼び出す機能をヒストリー機能と言います。「↑」や「↓」で使用できます。同じような長いコマンドを何回も打ち込む手間が省けます。
- 重要なコマンドをまとめると、以下のようになります。

`pwd`	現ディレクトリー (カレントディレクトリー) のパス (位置) の表示
`ls`	ファイルとディレクトリーの表示
`cd`	ワーキングディレクトリーの移動
`mkdir`	ディレクトリーの作成
`rmdir`	空のディレクトリーの削除
`cp`	ファイルやディレクトリーの複製
`mv`	名前変更や移動
`rm`	ファイルやディレクトリーの削除
`cat`	ファイルの表示や連結
`more`	ファイルの内容を一画面単位で出力
`man`	コマンドのオンラインマニュアル
↑又は↓	history。以前のコマンドの表示を行う。編集可能である。
[ctrl]+c	プロセスの強制終了
[Tab]	補完機能

2 コンパイルと実行

- C 言語のプログラムが書かれたソースファイルには、「hogehoge.c」のように拡張子「.c」が必要です。

- C 言語のソースファイル「hogehoge.c」をコンパイルして、実行ファイル「hugahuga」を作成するコマンドは、次の通りです。

  数学関数が無い場合　`cc -o hugahuga hogehoge.c`
  数学関数が有る場合　`cc -lm -o hugahuga hogehoge.c`

- ターミナルに「実行ファイル名 (例えば、hugahuga)」を打ち込んで [Enter] キーを押せば、プログラムは実行されます。

3　C 言語

今後、C 言語を用いての数値計算を学習する上で、C 言語の重要な事項をまとめます。

3.1　C 言語の基礎

- C 言語では、大文字と小文字は、区別されます。変数名 hogehoge と Hogehoge、hoGehoge は異なります。

- コメント文は、プログラムの内容をわかりやすくするために記述するものです。これは、人間のためのもので、コンパイラーは無視します。/*〜*/で囲まれた部分が、コメント文です。行をまたいでも、それは有効です。

- 識別子とは、変数、記号定数、関数などにつける名前のことです。名前に用いることができる文字は決まっています。英大文字「A〜Z」と英子文字「a〜z」、数字の「0〜9」とアンダースコアー「_」です。

- C はフリーフォーマットで記述できますので、文の区切りの記号が必要です。その区切りの記号にセミコロン「;」を用います。

3.2　データの型

- 変数は定義してから用いなくてはなりません。型を指定することにより、変数を定義できます。これは、コンパイラーがプログラムを実行するときに、必要な領域を確保するためです。

- 使用頻度が高い型は、以下の通りです。

型名	型指定子	変数宣言例
文字型	`char`	`char a, b;`
整数型	`int`	`int i,j;`
倍精度実数型	`double`	`double x, y;`

- C 言語では、変数の適用範囲は厳密に決められています。自動変数と外部変数があり、適用範囲が異なります。

ローカル変数	関数の中で定義され、その関数の中だけで使用できる。関数がコールされるとメモリー上に変数が配置される。その関数の処理が終わるとその変数は消滅する。通常良く使うのはこれである。
グローバル変数	関数の外で定義され、どの関数でも使用できる。プログラムが起動されるとメモリー上に変数が配置される。プログラムが終了するまで、変数は維持される。

3.3 制御文

- 通常プログラムは、上から下へと実行されます。しかし、実行の流れを変えたい場合が多々あります。そのプログラムの実行の順序を制御するのを制御文といいます。
- 使用頻度の高い制御文は、次の 3 個です。

1. `if(`条件 1`){` 文 1`}else if(`条件 2`){` 文 2`}else{` 文 3`}`
   - 条件 1 が真の時、文 1 が実行されます。条件 1 が偽の場合、次の条件 2 の真偽を判断し、真ならば文 2 を実行します。`else if` 文はいくらでも書くことができます。
   - 最初に真である条件に続く文を実行すると、`if` 文から抜けます。
   - 全ての `if` 又は `else if` の条件が偽ならば、`else` の文 3 を実行します。
2. `for(`初期値; 継続条件式; 再設定式`){` 文 `}`
   - 実行順序は、以下の通り。
     (1) 初期値の設定
     (2) 継続条件が真ならば、続く文を実行し、偽ならば `for` 文は終了する。
     (3) 再設定式を実行
     (4) 再び、(2) から実行する。
3. `do{` 文 `}while(`式`);`
   - 実行順序は、以下の通り。
     (1) 文を実行
     (2) 条件式が真ならば (1) へ戻り、偽ならば `do` 文は終了

3.4 配列

- 配列は、同じようなデータが多くある場合に使います。多くのデータに一つずつ名前をつけると大変です。1 万個のデータがあった場合、1 万個の名前を付けた変数を用意しますか?。下の例で、変数を用いての大量のデータ処理が不可能ということが分かるでしょう。
  - 1 万個の変数で領域を用意する場合の宣言

```
double aaa, aab, aac, aad, aae, aaf;
                    ⋮
double oun, ouo, oup, ouq;
```

– 配列で 1 万個の領域を用意する場合の宣言

```
double a[10000]
```

- 配列を使う場合も宣言が必要です。宣言の例は、以下の通りです。

配列の次元	要素数	宣言例
1 次元	100	`double x[100]`
2 次元	100×100	`double x[100][100]`
3 次元	100×100×100	`double x[100][100][100]`

- 配列添字は 0 から始まります。したがって、「`double x[1000]`」 と宣言した場合、使える配列は、`x[0]`〜`x[999]` までです。

- 添字である数字でデータの指定ができるため、メモリからのデータの読み書きが単純化できます。

3.5 関数

- C 言語は、関数の集まりです。その中で main 関数は特別で、そこから実行されます。

- プログラマーが関数を作成する場合、以下のように記述します。ここでは、hogehoge がプログラマーが作成した関数です。

```
 1  #include <stdio.h>
 2
 3  l^@hogehoge(^O);
 4
 5  /*-----------  main function ----------------------*/
 6  int main(){
 7    ;
 8    a = hogehoge()
 9    ;
10    return(0);
11  }
12
13  /* ---------- user defined function ----------------*/
14  l^@hogehoge(^O){
15    ;
16    return();
```

```
17  }
```

- 1 行目が関数のプロトタイプ宣言です。関数名、戻り値や引数の型を宣言しています。
- 8 行目で関数 hogehoge を呼び出しています。
- 関数 hogehoge の定義は、14〜17 行で行っています。
- 関数の戻り値は、16 行目の式の値です。

- 関数へのデータの渡し方に、2 種類あります。

  値渡し　　呼び出す側と呼ばれる側の関数が各々変数を用意します。仮引数は、実引数のコピーとなります。呼ばれた関数が処理をしても、呼び出した側の実引数の変数は、影響がありません。

  アドレス渡し　呼び出す側の実引数は、アドレスです。呼ばれる側は、ポインターを用意して、実引数のアドレスを受け取ります。呼ばれた関数が処理をすると、呼び出した側の実引数の変数にも影響があります。

3.6　ファイル処理

- ファイルへのデータの書き出しの手順は、以下のとおりです。

```
1  #include <stdio.h>
2  main(){
3    FILE *fp;
4
5    ;
6    fp = fopen("hogehoge","w");
7    fprintf(fp, "%e", a);
8    ;
9
10   fclose(fp);
11   return(0);
12 }
```

- 3 行目で、変数 fp をファイルポインターとして宣言しています。
- 6 行目で、ファイル hogehoge を書き込みモードで、オープンしています。
- 7 行目で、変数 a の値を指数形式 (%e) でファイルポインター fp が示すファイルに書き込んでいます。
- 10 行目でファイルをクローズしています。

4 プログラム例

4.1 Hello World

- おなじみの超基本文

```
1  #include <stdio.h>
2  int main()
3  {
4    printf("Hello World!!\n");
5    return(0);
6  }
```

4.2 繰り返し (while)

- 1〜100 の和を求めます。

```
1  #include <stdio.h>
2  int main()
3  {
4    int a, b;
5
6    a = 1;
7    b = 0;
8
9    while(a<=100){
10     b += a;
11     a++;
12   }
13
14   printf("b = %d\n",b);
15
16   return(0);
17 }
```

4.3 繰り返し (for 文)

- 1〜100 の和を求めます。

```
1  #include <stdio.h>
2  int main()
```

```
 3  {
 4    int a, b;
 5
 6    a = b = 0;
 7
 8    for(a=1; a<=100; a++){
 9      b += a;
10    }
11
12    printf("b = %d\n",b);
13
14    return(0);
15  }
```

4.4 制御 (if 文)

- 1〜100 の和を求めます。

```
 1  #include <stdio.h>
 2  int main()
 3  {
 4    int a, b;
 5
 6    a = b = 0;
 7
 8   next: b+=a;
 9
10    if(a<100){
11      a++;
12      goto next;
13    }
14
15    printf("b = %d\n",b);
16
17    return(0);
18  }
```

4.5 関数 (値渡し)

- 5+6 の計算を関数 sum で計算しています。

```
 1  #include <stdio.h>
 2  int sum(int x, int y);
 3
 4  int main()
 5  {
 6    int a,b,wa;
 7    a=5;
 8    b=6;
 9
10    wa=sum(a,b);
11
12    printf("%d+%d=%d\n",a,b,wa);
13
14    return(0);
15  }
16
17  int sum(int x, int y)
18  {
19    return(x+y);
20  }
```

4.6 関数 (アドレス渡し)

- 関数 swap によって、実引数の値を入れ替えています。アドレス渡しであるため、関数側での操作が実引数に反映されます。

```
 1  #include <stdio.h>
 2  void swap(int *a, int *b);
 3
 4  int main()
 5  {
 6    int a, b;
 7
 8    a=1;
 9    b=-1;
10
11    swap(&a,&b);
12    printf(" %d  %d\n",a,b);
13
14    return(0);
```

```
15  }
16
17  void swap(int *a, int *b)
18  {
19    int c;
20
21    c=*a;
22
23    *a=*b;
24    *b=c;
25  }
```

4.7 ファイル処理 (データ書き出し)

- ファイルに変数の値と三角関数の値を書き出しています。

```
 1  #include <stdio.h>
 2  #include <math.h>
 3
 4  int main()
 5  {
 6    FILE *out;
 7    double pi=4*atan(1.0);
 8    double theta, s, c, t;
 9    int i;
10
11    out = fopen("calresult","w");
12
13    for(i=0;i<=100;i++){
14      theta = i*pi/100;
15      s = sin(theta);
16      c = cos(theta);
17      t = tan(theta);
18      fprintf(out,"%f\t%f\t%f\t%f\n",theta, s, c, t);
19    }
20
21    fclose(out);
22
23    return(0);
24  }
```