補講のテキスト

山本昌志*

2004 年 6 月 22 日

再試験に向けて、今まで学習した範囲をまとめる。このプリントの内容を十分理解して、中間テストに臨むこと。合わせて、中間テストの解答は十分に理解すること。

1 用語

教科書の第 1 章で、重要な記述を、以下に示す。これらは、確実に内容を理解し、憶える必要がある。

- コンピュータに何をどのようにさせるかを記述したものをプログラム (program) と呼ぶ。(p.13)
  - コンピューター機材を「ハードウェアー」と呼ぶ
  - コンピューターに仕事をさせるためのプログラムを「ソフトウェアーと呼ぶ」
- コンピュータが理解できる言葉は、機械語 (マシン語) だけである。それは、「$\cdots 0010110100110 \cdots$」のように、0 と 1 の数字が並んだものである (p.16)。これを、人間が理解するのは大変です。
- そこで、人間がわかりやすい言語 (たとえば C 言語) でプログラムを書き、それを翻訳して機械語に直すことが考えられた。翻訳のように単純作業はコンピューターが得意である。
- コンピューターに与える指示を書いたものがプログラムである。指示を与える言葉をプログラミング言語と言い、いろいろな種類がる。しかし、どの言葉でも最終的には機械語に翻訳して、コンピューターは指示通りに仕事をする。
- コンピュータは単純な仕事しか理解できないので、目的達成までの方法や道筋を細かく表現して、指示を与える必要がある。しかも、正確に伝えなければ、コンピュータは正しく仕事をしない。(p.20)
- 問題解決までの道筋のことをアルゴリズム (algorithm) と言う。(p.26)
- コンピュータに仕事をさせるには、人間が少しでも書きやすい (理解しやすい) ようなプログラミング言語を使ってプログラムを作り、それをコンピュータがわかるように翻訳し、コンピューターでそのプログラムを動かす手順をとる。(p.30)
- C 言語は、翻訳という手続きを必要とするプログラミング言語である。このような言語は、共通したプログラム構築の流れがる。(p.30-36)

*国立秋田工業高等専門学校　電気情報工学科

1. プログラムを記述 (コーディング)
   - 問題を解くための手順 (アルゴリズム) を、プログラム言語で決められた文法で記述する。この作業をコーディングと言う。
   - コーディングの作業により記述されたものをソースプログラム、あるいはソースコードと呼ぶ。
   - 通常、ソースプログラムは、文字を入力しテキストファイルで保存できるテキストエディタで書く。
2. プログラムを翻訳 (コンパイルとリンク)
   - ソースプログラムを機械語に翻訳する作業をコンパイルとリンクと言う。
   - コンパイルはコンパイラーと言うソフトウェアーにより、ソースプログラムをオブジェクトプログラムに変換する。
   - リンクはリンカーと言うソフトウェアーにより、オブジェクトプログラムとライブラリーから、機械語の実行プログラムを作ります。
   - 実行プログラムが記述されているファイルを実行ファイルと言う。
3. プログラムの動作確認
   - UNIX の場合、実行ファイル名をタイプすることにより、プログラムを実行させます。
4. プログラムの誤りの修正 (デバッグ)
   - ソースプログラムの翻訳時や機械語のプログラムの実行時の誤りを修正する作業をデバックと言う。

2 UNIX

2.1 ファイル構造

- UNIX のハードディスク[1]のファイル構造は、図 1 のようにツリー (木) 構造呼ばれる階層構造になっている。それは、ファイルとディレクトリーから構成される。
- ハードディスクなどに記録されたデータのまとまりをファイルと言う。コンピュータが実行することができる命令の集合であるプログラムファイルと、コンピュータの利用者が作成した情報を記録しておくデータファイルがある。
- ファイルを分類・整理するための保管場所をディレクトリー[2]と言う。関連する複数のファイルをまとめて一つのディレクトリーに入れることにより、効率的に記憶装置を管理することができる。ディレクトリーの中にさらにディレクトリーを作成することもでき、階層構造によって細かい分類を表現することもできる。
- ディレクトリーには、以下のように表現されるものがある。

[1] 実際はハードディスクに限らず、CD-ROM やフロッピーディスクも、このツリー構造に含まれる。
[2] Mac や windows ではフォルダーと呼ぶ

- 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う。[3]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う。2つのピリオド「..」が、親ディレクトリーを表す。
- カレントディレクトリーの1つ下のディレクトリーをサブデレクトリーと言う。複数存在することが可能なので、それぞれの名前で表すしかない。
- 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う。

- ファイルやフォルダの所在を示すものをパスと言う。ファイルやフォルダのハードディスクでの住所みたいのものである。それは、ディレクトリー名を書き並べることにより表すことができ、その区切りには「/」(スラッシュ) を使います。
- パスの表し方は、2通りある。最上位のルートディレクトリー「/」から表す絶対パスと、カレントディレクトリーから表す相対パスである。例えば、図1の yamamoto をカレントディレクトリーとし、prog.c のパスは、

  絶対パス　`/home/stu/e/work/prog.c`
  相対パス　`../../stu/e/work/prog.c`

  となる。

2.2 UNIX のコマンド

- 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`」である。
- 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`」である。
- ディレクトリーを移動する場合コマンド「`cd`」を使う。以下のような使い方がある。
  - 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。
  - サブディレクトリー hogehoge に移動するコマンドは、「`cd hogehoge`」です。
  - ホームディレクトリーに移動するコマンドは、「`cd`」です。
  - 相対パスや絶対パスを指定して、移動することもできる。例えば、図1の yamamoto をカレントディレクトリーとし、hogehoge に移動するには、

    絶対パス　`cd /home/stu/e/work/hogehoge`
    相対パス　`cd ../../stu/e/work/hogehoge`

    とする。

[3]ワーキングディレクトリーと言うこともある。

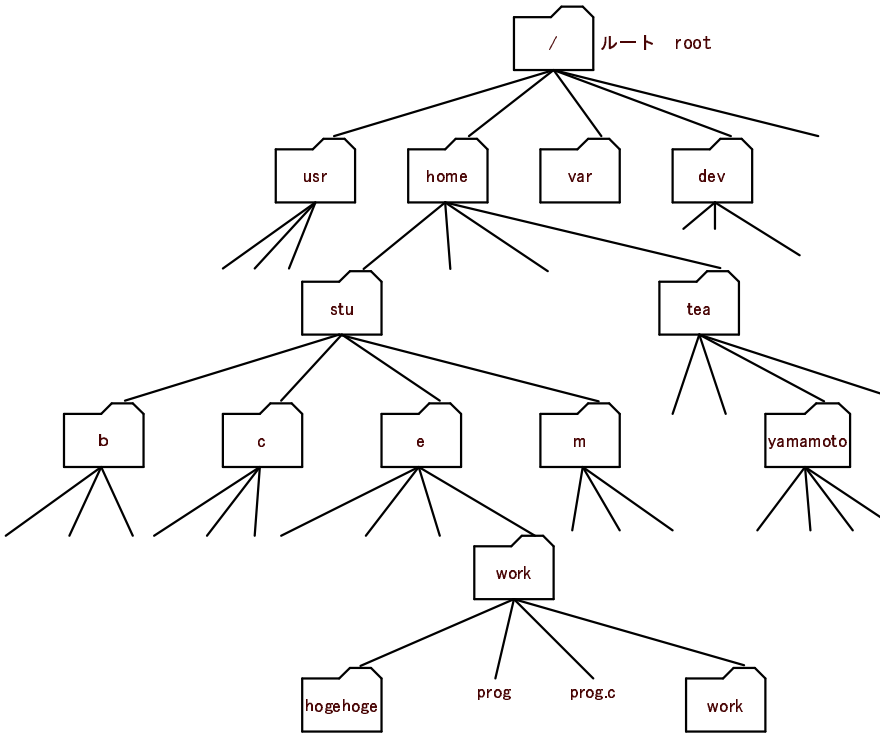

図 1: UNIX のファイルの構造 (秋田高専の namahage)

- 新規にディレクトリー hogehoge を作るコマンドは、「`mkdir hogehoge`」である。
- 空っぽのディレクトリー hogehoge を削除するコマンドは、「`rmdir hogehoge`」である。
- ファイルやディレクトリーを削除するコマンドは、「`rm`」である。
  - ファイル hogehoge を削除する場合、「`rm hogehoge`」とする。
  - 中身にあるディレクトリー hugahuga を削除する場合、「`rm -rf hugahuga`」とする。-rf はオプションと呼ばれるもので、r は recursive (再帰) の略で、f は force(強制) の略である。
- 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

2.3 コンパイル・リンクと実行

- 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」が必要である。
- C 言語のソースファイル「hogehoge.c」をコンパイル・リンクして、実行ファイル「hugahuga」を作成するコマンドは、次の通りである。

```
cc -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 言語の文法

3.1 基本

3.1.1 約束事

- 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。
- コンピューター内部では、\(バックスラッシュ) と ¥ (円マーク) の取り扱いは全く同じです。
- 空白、改行、タブは、関数の間に好きなだけ入れることができる。

- { と } は対応しており、{ と } で囲まれた部分は、一つの処理のまとまりを表す。
- ひとつの文は、;(セミコロン) で終わらなくてはならない。これは、日本語の文末につける「。」に相当する。
- 足し算は+、引き算は-、掛け算は*、割り算は/で表すことができ、( ) を使って、数学と同じように計算式を書くことができる。
- 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。
- プログラムは main() 関数から実行される。
- 基本的には、プログラムは上から下へと処理の動作が行われる。

3.1.2 おまじない

- 当面、プログラムの最初と最後にか書かれる、以下の部分はおまじないとして憶えておきます。もう少し進んだときに、その内容は教えます。

```
#include <stdio.h>

int main()
{

   文;

   return(0);
}
```

3.2 printf() 関数

- ディスプレイ[4]に、"(ダブルクォーテーション) で囲まれた文字列や数値などの表示と、改行などを指示する。

C 言語の文	出力	説明
printf("Hello world");	Hello world	表示後、改行なし
printf("Hello world\n");	Hello world	表示後、改行あり。

- 改行したい場合は、改行したい場所に「\n」を記述する。

[4]正確には標準出力

C 言語の文	出力	説明
`printf("Hello\nworld\n",i);`	`Hello` `world`	Hello の後と world の後に改行

- 変数を表示する場合、その数値や文字は、ダブルクォーテーションの間で、Y_n で与えられる変換仕様を記述する。変換仕様は対応する変数の出力方法を決めるものである。
- Y_n に対する変換仕様は、以下の通りである。

整数	`%d`
実数値	`%f`
より精度の高い実数値	`%lf`
1 文字	`%c`

今回のテスト範囲では、`%d` と`%f` が使えればよい。i=-1234、x=-1.234 の場合、次のようになる。

C 言語の文	出力	説明
`printf("i=%d",i);`	`i=-1234`	
`printf("x=%f",x);`	`x=-1.23000`	

- 文字や数字を表す場合、その文字数をフィールド幅と言う。
- 整数表示のフィールド幅を調整するときは、変換仕様`%d` の`%`と `d` の間に、最小フィールド幅を記述する。整数の桁よりもフィールド幅が少ない場合は、必要分が追加される。また、フィールド幅の方が多い場合は、右詰で左側に空白が挿入される。たとえば、i=-1234 とした場合、以下のようになる。

C 言語の文	出力	説明
`printf("%d",i);`	`-1234`	
`printf("%3d",i);`	`-1234`	表示のフィールドが不足なので追加
`printf("%4d",i);`	`-1234`	表示のフィールドが不足なので追加
`printf("%5d",i);`	`-1234`	
`printf("%6d",i);`	` -1234`	左側に 1 個の空白あり

- 少数部の桁を調整するときは、変換仕様`%lf` の`%`と `lf` の間に、「最小フィールド幅. 小数部の桁数」と記述する。表示されない桁は、四捨五入される。たとえば、x=-1.234 とした場合、以下のようになる。

C 言語の文	出力	説明
`printf("%lf",x);`	-1.234000	
`printf("%.1lf",i);`	-1.2	
`printf("%.3lf",i);`	-1.234	
`printf("%3lf",x);`	-1.234000	
`printf("%3.1lf",i);`	-1.2	表示のフィールドが不足なので追加
`printf("%3.3lf",i);`	-1.234	表示のフィールドが不足なので追加
`printf("%8lf",x);`	-1.234000	表示のフィールドが不足なので追加
`printf("%8.1lf",i);`	-1.2	左側に 4 個の空白あり
`printf("%8.3lf",i);`	-1.234	左側に 2 個の空白あり

3.3 変数

- 変数は、値を入れておく箱のようなものである。1 つの変数に 1 個の値を入れておく (記憶) ことができる。
- いろいろな変数の型があるが、以下のものがよく使われる。

変数の型名	変数の取り得る値
`int`	整数
`float`	実数
`double`	精度の高い実数

  今回のテスト範囲では、`int` と `double` が使えればよい。

- 変数を使う場合、プログラムの最初に変数の宣言を行う必要がある。たとえば、整数変数 i と max_i、倍精度実数変数 x と max_x を使う場合は、次のように宣言する。

```
int i, max_i;
double x, max_x;
```

- 変数に値を入れるためには、代入演算子 (=) をつかう。これは、右辺の計算式の値を、左辺の変数に代入する。したがって、右辺と左辺を入れ替えてはいけない。

```
i=123;
x=3.0*i+1.234;
```

3.4 scanf() 関数

- キーボードから、値 (文字や数値) を読み込むために、`scanf()` 関数を使う。読み込んだ値は、変数に格納される。
- `scanf()` 関数の記述の仕方は、次の通りである。

```
scanf("変換仕様の並び ", &変数名, &変数名, ···, &変数名);
```

- 変換仕様とは、キーボードから入力されたものがどのような値か判断するために使う。主なものは以下の通りである。

整数	%d
実数値	%f
倍精度実数値	%lf
1 文字	%c

- 変数名は、先頭に&をつける必要がある。

- キーボードから実数変数を読み込み、その値を変数 x に代入する。さらに、キーボードから整数変数を読み込み、その値を i に代入する。このプログラムは、以下のように書く。

```
scanf("%lf",&x);
scanf("%d",&i);
```

4　プログラム例

次のプログラムの内容を理解し、書けるようになっておくこと。

4.1　メッセージの表示

```
1  #include <stdio.h>
2
3  int main()
4  {
5    printf("Hello.\n\n");
6    printf("How are you.\n");
7
8    return(0);
9  }
```

4.2　携帯電話の料金

```
1  #include <stdio.h>
2
3  int main()
4  {
5    int base, time, number;
6
7    base = 3000;
8
9    printf("通話時間を入力してください [分単位]\n");
10   scanf("%d", &time);
11
12   printf("メール送信数を入力してください\n");
13   scanf("%d", &number);
14
15   printf("----- 使用料金 -------\n");
16   printf("%d 円\n", base+10*time+5*number);
17
18   return(0);
19 }
```