C言語の基本的な知識・定数

山本昌志*

2006年4月18日

概 要

前回の授業に引き続き，C言語でプログラムを作成するときの基本的なこと述べる．始めに，UNIXでプログラムを楽に作成するためのテクニックを示す．引き続き，教科書の1章と2章の大事なことに付いて説明する．ここに書かれている全ての内容を説明する時間もないので，今後の数値計算の講義で必要となる項目にしぼって説明する．

1 プログラムを楽に作成する方法

以下のようなテクニックを身につけると，プログラムの作成が容易になる．

- 似たようなプログラムをコピーと修正により，新規に作成する．毎回，最初からプログラムを作成すると，手間が大変だし，時間もかかる．以前作成したプログラムのソースファイルをコピーして，修正する方が断然，楽である．
- プログラムは，ディレクトリー毎にわけて管理する．これから，諸君は多くのプログラムを作成することになる．作成したプログラムが，すぐに利用できるように整理しておくと便利である．良いプログラムは，財産となる．少なくとも，卒業研究に利用することは可能である．いざ使うときに，すぐにみつからないと利用する気も失せてしまう．
- コマンドをタイプするCUIでのファイル操作が性に合わなければ，GUIのファイル操作を使えば良い．
- 補完機能を上手に使うとタイプする回数が少なくなり，プログラマーのストレスが減る．長いファイル名を入れるときに，途中までタイプして，[Tab]キーを押すと，それ以降のファイル名を補ってくれる．ファイル名を最後まで入れる必要はない．また，コマンドも補ってくれる．
- コピー・ペーストを利用しよう．一番単純なコピーペーストは，マウスの左ドラッグと中クリックである．これは，emacsに限らず，いつでもどこでも使える．
- history機能を利用しよう．UNIXを使ってプログラムを作成しているとき，同じようなコマンドを何回もタイプする．例えば，`gcc -o hoge fuga.c`などである．これをいちいちタイプしていたら，それだけでプログラム作成の意欲が失う．このようなときは，[上矢印]や[下矢印]を使うと，良い．以前タイプした文字が出てくる．編集も可能である．

*独立行政法人秋田工業高等専門学校電気工学科

2　基本事項

ここでは，教科書 [1] の第一章の内容で，重要なことを述べる．

2.1　大文字と小文字の区別 教科書 p.7

2.1.1　動作と内容

C 言語では，大文字と小文字は，区別される．その例をリスト 1 に示す．このプログラムは，以下のように動作する．このプログラムのうち，実行される文は，6〜12 行目である．そのほかの文は，プログラムを動作以前に必要な文で，プログラムがメモリーに格納されるまでに，その仕事は終わっている．

1. 変数 hogehoge に 1 を格納する．
2. 変数 Hogehoge に 2 を格納する．
3. 変数 hoGehoge に 3 を格納する．
4. hogehoge の値をディスプレイに書き出す．hogehoge = 1 と表示される．
5. Hogehoge の値をディスプレイに書き出す．
6. hoGehoge の値をディスプレイに書き出す．

このプログラムでは，3 つの異なる変数名 hogehoge と Hogehoge，hoGehoge が使われている．おなじ「ほげほげ」であるが，アルファベットの大文字と小文字が異なっている．C 言語では大文字と小文字は，区別され，異なった変数としてあつかわれる．これは，プログラムの実行結果を見れば分かる．

リスト 1: 大文字と小文字を区別することを示すプログラム例

```
1  #include <stdio.h>
2
3  int main(void){
4    int hogehoge, Hogehoge, hoGehoge;
5
6    hogehoge = 1;
7    Hogehoge = 2;
8    hoGehoge = 3;
9
10   printf("hogehoge = %d\n",hogehoge);
11   printf("Hogehoge = %d\n",Hogehoge);
12   printf("hoGehoge = %d\n",hoGehoge);
13
14   return 0;
15 }
```

実行結果

```
hogehoge = 1
Hogehoge = 2
hoGehoge = 3
```

2.1.2 ソースプログラムの説明

このプログラムのソースの内容は，次のようになっている．リスト 1 の各行毎にその役割を示す．現時点で，このプログラムがかけるようになる必要がある．

- `#include <stdio.h>`は，当面おまじないだと思って欲しい．
- `int main(void)` もおまじない．この後の中括弧 { } の中身が `main` と言う関数の本体である．当面は，この中に書かれたものが実行されると考えて欲しい．
- `int hogehoge, Hogehoge, hoGehoge;` で 3 つの整数型の変数を用意している．変数とは，データを記憶する場所に名前が付いたものである．アセンブラでは，アドレスを指定することでデータにアクセスしたが，C 言語では変数名を使う．当然，この変数名はメモリーの特定のアドレスを表す．数学の変数と似たような使い方をするが，メモリーとの関係を考えると，その意味はかなり違ったものになる．また，C 言語に変数には型というものがあり，記憶することができるものが決まっている．ここで用意した，3 つの変数 (ほげほげ) は整数しか記憶できない．
- `printf()` 関数で変数のデータを書き出している．`printf` は括弧内に従いディスプレイ[1]に表示せよという命令である (図 1 を参照)．括弧内のダブルクォーテーション[2]で囲まれた部分を表示する．ここの，`%d` は出力変換仕様 (教科書 p.320〜) と言い，`hogehoge` の値を 10 進数 (decimal) で表示する．`\n` は改行を表す．これがあるとディスプレイ上で改行される．
- `return 0`．これもおまじない．

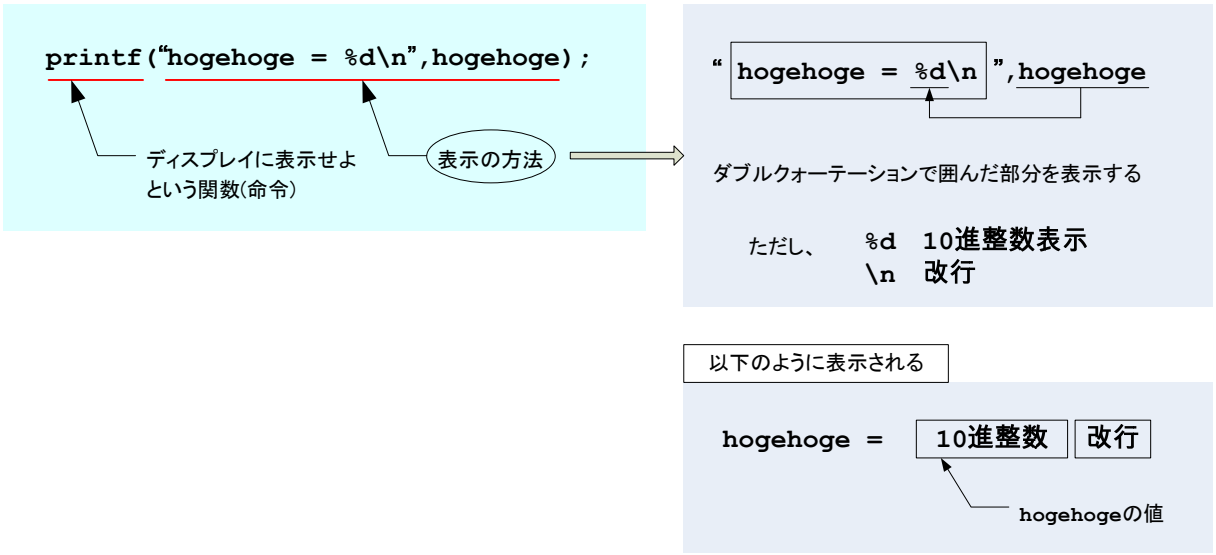

図 1: `printf()` 関数の説明.

[1]正確には，標準出力.
[2]記号 ”をダブルクォーテーションと言う.

おまじないがかなりある．諸君は，まだ C 言語の学習をはじめたばかりなので，その内容まで理解しようとすると，先に進むことができない．多少のことは目をつぶって，意図した通りに動作するプログラムを作ることに専念すべきである．

おまじないと動作を記述する部分は，図 2 のように書く．プログラムを作成するときには，最初にこのおまじないを書く．つぎに，動作の部分を書くようにすればプログラムはできあがる．なにはともあれ，このおまじないにはご利益はある．

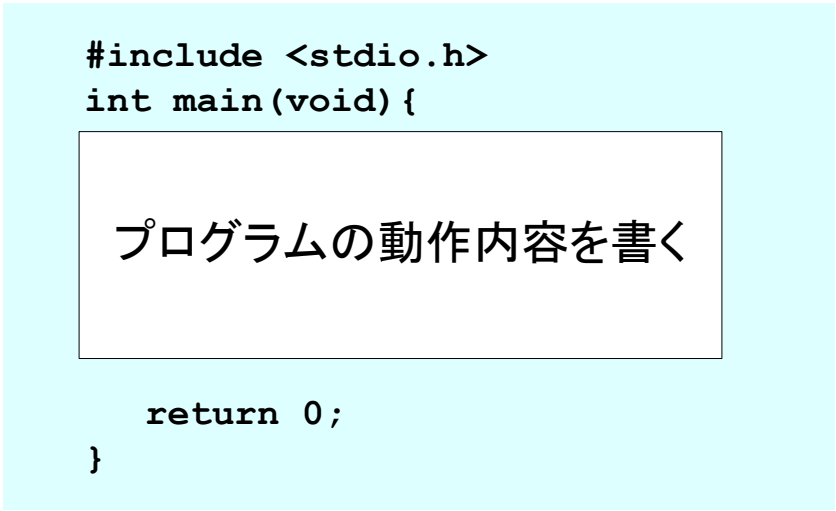

図 2: C 言語のプログラムの構造．おまじないの部分と動作を記述する部分．

2.2 注釈 (コメント文) 教科書 p.11

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ある．これは，人間のためのもので，コンパイラーは無視する．プログラムを維持・管理するときの参考に用いる．良いプログラムは，コメント文が大量に書かれている．FORTRAN の場合，第一カラムが”*”，または”C”の場合，その行はコメント文となるのは，以前，学習したとおりである．C 言語の場合は，/*〜*/で囲まれた部分が，コメント文となる．行をまたいでも，それは有効である．

ANSI[3]の規格に反するが，//をコメント文の開始として使える．この 2 つのスラッシュから，行末までがコメントとなる．

リスト 2 にコメント文が入ったソースプログラムを示す．また，リスト 3 には，コメント文が無いプログラムを示す．どちらもはまったく同じ実行結果になる．それどころか，コンパイルしてできた実行ファイル (機械語) も同一である．このことからも，コンパイラーはコメント文を無視することが分かる．

リスト 2: コメントの書き方

```
1 /* ========================================== */
```

[3]American National Standard Institute．アメリカ規格協会と呼ばれ，日本の JIS 見たいなもの．ANSI で決められた C 言語が標準とされている．

```
2  /* ==     円の面積の計算                                  */
3  /* ============================================== */
4  #include <stdio.h>
5
6  int main(void){
7    double pi;
8    double r, s;
9
10   pi = 3.141592;          /* 円周率 */
11   r = 1.0;                /* 円の半径 */
12   s = pi*r*r;             /* 円の面積
13                               次の行にまたがっても良い */
14
15   printf("s = %f\n",s);   // ANSIの標準ではないが, これもOK
16
17   return 0;
18
19 }
```

実行結果

```
s = 3.141592
```

リスト 3: コメントの無いソースプログラム

```
1  #include <stdio.h>
2
3  int main(void){
4    double pi;
5    double r, s;
6
7    pi = 3.141592;
8    r = 1.0;
9    s = pi*r*r;
10
11
12   printf("s = %f\n",s);
13
14   return 0;
15
16 }
```

実行結果

```
s = 3.141592
```

2.3 識別子 教科書 p.12

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ある．名前に用いることが出来る文字は決まっており，以下のとおりである．

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r t t u v w x y z
0 1 2 3 4 5 6 7 8 9 _（下線，アンダースコアー，アンダーバー）

2.4　全角空白と半角空白 教科書 p.13

全角の空白と半角の空白は，まったく異なる．全角の空白は，日本語を表示する時以外，使ってはならない．全角の空白を書くと，非常に分かりにくいバグの原因となることがあるので極力使わない方がよいであろう．

2.5　フリーフォーマット 教科書 p.13

FORTRAN は，7 カラム目から，プログラムは記述するという約束がる．しかし，C には，どこからでも，プログラムを書くことが出来る．そのため，わかり易いプログラムを書くために字下げを使うことができる．emacs の場合，行の先頭で [Tab] を打つと自動的に字下げをしてくれる．字下げを上手に使って，わかりやすいプログラムを書くことが，上達の早道である．

リスト 4 に字下げをして記述したプログラムを示す．プログラムの構造が分かり易くなっていることに気づくだろう．このプログラムでは，`for` 文を使って，繰り返しの構造が使われている．`i` の値を変えなら，`for` の後の中括弧 { } の中を 9 回繰り返している．この繰り返し (ループ) の部分を字下げして分かり易くしている．

リスト 4: 字下げの例

```
1  #include <stdio.h>
2
3  int main(void){
4    int i;
5
6    for(i=1; i <= 9; i++){
7      printf("%d: Hello World !!\n",i);
8    }
9
10   return 0;
11 }
```

実行結果

```
1: Hello World !!
2: Hello World !!
3: Hello World !!
4: Hello World !!
5: Hello World !!
6: Hello World !!
7: Hello World !!
8: Hello World !!
9: Hello World !!
```

このプログラムでは，以下の繰り返し文 (ループ) が使われている．

```
for(i=1; i <= 9; i++){
  printf("%d: Hello World !!\n",i);
}
```

これは，

- i の値を 1 から，　　i=1
- 9 まで，　　i<=9
- 1 ずつ増加させて，　　i++
- 中括弧 {　　} の内部を繰り返せ

と解釈する．だから，先頭の数字の値が変化しながら，Hello World !!が 9 回繰り返されたのである．

ここで，少し printf 文と繰り返し文の練習をしてみよう．

[**練習 1**] リスト 4 を直して，以下の出力をするプログラムを作成せよ．

```
1 ==== Hello World !! ==== 1
2 ==== Hello World !! ==== 2
3 ==== Hello World !! ==== 3
          .
          .
          .
100 ==== Hello World !! ==== 100
```

2.6　セミコロン

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ある．その区切りの記号にセミコロンを用いる．

3　定数 (教科書の 2 章)

教科書の p.20 を見ると分かるようにいろいろな定数がある．しかし，この講義は主に数値計算について学習するので，整数定数と浮動小数点定数がほとんどである．文字定数や文字列リテラルは使用頻度が少ない．その他のものはほとんど使われない．

3.1　整数型と実数型

整数型と実数型の定数を変数に代入して，画面へ出力するソースをリスト**??**に示す．各行の内容は以下の通りである．

1, 3, 12, 13 行 先ほどのべたように，これはおまじない．

4 行 整数型の変数 seisu を宣言．詳細は次回の授業で述べる．

5 行 倍精度実数型の変数 jisu を宣言．詳細は次回の授業で述べる．

7,8 行 変数に定数を代入．コンピューター言語で値を代入する場合，左辺の変数に右辺の計算結果を代入することになる．必ず，左辺は変数で，右辺は数値となる．

10 行 ダブルクォーテーションで囲まれた部分中の%d の部分に変数 seisu の値を 10 進数 (decimal) で，%e の部分に変数 seisu を e タイプで置き換えディスプレイに表示する．この%d や%e を変換仕様という (教科書 p.322)．\n は改行である．

リスト 5: 定数の学習プログラム

```
1  #include <stdio.h>
2
3  int main(void){
4    int seisu;
5    double jisu;
6
7    seisu = 65;
8    jisu = -69.53e-7;
9
10   printf("seisu = %d     jisu = %e\n", seisu, jisu);
11
12   return 0;
13 }
```

実行結果

```
seisu = 65    jisu = -6.953000e-06
```

リスト 5 を直して，以下の練習問題を実行させよ．

[**練習 1**] 変数 seisu に -1234 を jisu に -6.987×10^{-68} を代入するプログラムを作成せよ．

[**練習 2**] 変数 seisu に -6.987×10^{-68} を jisu に -1234 を代入するプログラムを作成せよ．そして，実行結果の内容を考察せよ．

[**練習 3**] 変数 seisu に $-10/3$ を jisu に $-10/3$ を代入するプログラムを作成せよ．そして，実行結果の内容を考察せよ．

3.2 エスケープシーケンス

教科書の表 2-4(p.28) のものをエスケープシーケンスと言う．これは 2 つあるいはそれ以上の文字列で表す特殊文字である．それらの機能は表に書いてあるとおりであるが，数値計算で重要なものは，\n と \t である．とりあえず，この 2 つの動作を理解せよ．

リスト 5 の 9 行目の printf 関数のダブルクォーテーション内を変えて，以下の練習問題を実行せよ．

[**練習 1**] \n を適当に挿入して，その動作を確認せよ．挿入は，1 個のみならず，2〜3 個それを挿入した場合も確認せよ．

[**練習 2**] \t を適当に挿入して，その動作を確認せよ．挿入は，1 個のみならず，2〜3 個それを挿入した場合も確認せよ．

[**練習 3**] Hello World !!のプログラムのダブルクォーテーション内のいろいろな場所に，\n と \t を書いて，その動作を理解せよ．

参考文献

[1] 林春比古. 新訂 C 言語入門 シニア編. ソフトバンク パブリッシング, 2004.