計算機工学II

竹村　裕（助教） E-mail:takemura@rs.noda.tus.ac.jp
川鍋　充（TA）　E-mail:mitsuru@me.noda.tus.ac.jp

第八回　2007年6月6日

1　C言語の基礎 1

1.1　コンパイル

コンパイルとは，プログラミング言語を用いて作成したソフトウェアの設計図(ソースコード) を，コンピュータ上で実行可能な形式 (オブジェクトコード) に変換することである．C 言語では，例えば，

```
nodat001% gcc -o filename filename.c
nodat001% ls
filename*   filename.c
```

などである．このプログラムを実行したければ，

```
nodat001% ./filename
```

とすればよい．下記の sample1.c は最も簡単なプログラムの例である．コンパイルして，実行しても何も起こらないが，動作環境があることは調べられる．

sample1.c

```
/*もっとも簡単なプログラム*/
main() {}
```

2　基本的な決まり

C 言語を書くには幾つかの決まりごとがある．基本的な構造は以下のようになる．

```
#include <?????>
関数の宣言
変数の宣言
関数群（実際のプログラム）
```

例えば，

sample2.c

```
/*プログラムの書き方*/
#include <stdio.h>
void fun();  /*関数の宣言*/
int a;       /*変数の宣言*/
main()       /*実際のプログラム*/
{
a=3;
printf("a=%d\n",a);
fun();
}
```

これを実行すると，

```
nodat001% ./samlpe2
a=3
```

2.1　予約語

自分で作る変数や関数などの名前は「識別子」と呼ぶ．識別子の名前は基本的にはプログラマが自由に命名できる．ただし次のような制約がある．

- 半角英数字および半角アンダーバ (_) のみ使用でき，最初の 1 文字は英字である
- 大文字，小文字は区別される

- 31 文字以内であること (古いコンパイラでは最初の 6 文字)

また, 慣用的に使われる決まりもありる. C 言語の上位である C++言語や, C++言語と (文法的に) 似ている Java 言語では当てはまらないのがやっかいなのですが … .

- 基本的に小文字を使う
- 標準関数と同じ名前は使わない
- 標準関数, 予約語の大文字, 小文字だけの違いのものは使わない

さらに, 以下の予約語は使用できない

```
auto   break   case   char   const
continue   default   do   double
else   enum   extern   float   for
goto   if   int   long   register
return   short   signed   unsigned
void   volatile   while
```

2.2 データの型と宣言

変数を使う場合には宣言をしなくてはならない. 何を宣言するか, どんなデータの型か, 名前は, 初期値はなど.(初期値は省略することがでる.) 宣言する場所が関数の外か (ブロックの外), 関数の中か (ブロックの中), で影響する範囲が代わる. 関数の外で宣言した場合, グローバル変数といい, そのファイルの中のすべての場所で使用することができる. ただしこれはあまり良い宣言方法ではない. なぜならば, 複数のプログラマーで大きいプログラムを作っている場合, 不正に書き換えられてしまう可能性があり, バグの発生につながる可能性がある. 特別な理由がある場合を除いて, 後者のブロックの最初で宣言するべきである.
C 言語では表:1 の型が用意されている. 例えば, 以下のような宣言になりる.

```
int a; /* 変数名「a」を int 型で宣言*/
char ch1,ch2;
/*変数名「ch1」と「ch2」を char 型で宣言*/
unsigned long l;
/*変数名「l」を unsigned long 型で宣言*/
```

3 入出力

3.1 printf(書式指定文字列, 変数, 変数…);

printf() の f は “format” (書式) の f です. printf() は書式指定を行うことにより, 同じ「65」という数値でも, 10 進数で出力したり, 文字で出力したりというように出力形式を変えることが出来る.

3.2 printf() のオプション

%	フラグ	フィールド幅	精度	変換指定文字

変換指定文字
出力したいデータの型にあわせて変換指定文字 (表:2 参照) を選択.

```
int a = 14;
double data = 0.0123;
char   str[] = "Hello";

printf("%d\n", a);
printf("%f %e\n", data, data);
printf("%s\n", str);
printf("error!!\n\a");
```

出力結果

```
14
0.012300 1.230000e-02
Hello
error!!
```

フラグ
変換指定には見栄えを良くするためにオプションをつけることができる. 表:3 によく使われるオプション指定を示す.

```
int a = 12, b = 58;
char str[] = "Hello";
```

```
printf("右詰め:%10s\n", str);
printf("左詰め:%-10s\n", str);
printf("符号あり:%+d\n", a);
printf(" 8 進表示:%#o\n", b);
printf("16 進表示:%#x\n", b);
```

出力結果
右詰め：　　Hello
指定がないと 10 文字分右詰め
左詰め：Hello
10 文字分左詰めで表示
符号あり：+12　　+ の符号を付加
 8 進表示：072　　先頭に 0 を付加
16 進表示：0x3a　　先頭に 0x を付加

フィールド幅
数値の出力幅を指定.

```
int data = 123;

printf("%d\n", data);
printf("%5d\n", data);
printf("%10d\n", data);
printf("%2d\n", data);
printf("%05d\n", data);
```

出力結果
123　　オプションなしのとき
  123　　スペース含めて 5 文字
　　123　スペース含めて 10 文字
123　　指定が小さい場合は必要幅
00123　　0 フラグがあると 0 を詰める

精度
実数の小数点以下の桁数を指定.

```
double x = 654.321;

printf("%f\n", x);
printf("%12f\n", x);
printf("%9.2f\n", x);
```

出力結果
654.321000　　オプションなしのとき
..654.321000
小数点を入れて 12 桁（小数点以下の桁は標準値）
...654.32
小数点を入れて 9 桁（小数点以下 2 桁）
（もちろん，”...”は表示されない！）

3.3 scanf(書式指定文字列, アドレス,…);

キーボードから書式付きで入力. scanf() の f は ”format”（書式） の f です. scanf() は printf() と同様に書式指定を行うことにより，「A」のキーを押しても，16 進数で入力したり，文字で入力したりというように入力形式を変えることが出来る.

書式指定文字列
表:4 に scanf の書式指定文字列を示す.
※ double 型の入力が “%lf”であることに注意!!

```
int    a;
double b;
char   c[100];

scanf("%d", &a);
scanf("%lf",&b);
scanf("%s",  c);
```

※ &が無いことに注意 !!

4 制御文

制御文とは，プログラムの流れを制御するもの. C 言語も他の多くのプログラム言語と同様，上から下にプログラムは実行される．このように上から下に実行される流れを「順次構造」とう．しかし，ある処理を繰り返したり，条件によって異なる処理をさせたい場合は，順次構造以外の流れが必要になる.

- 順次構造
- 分岐構造
  - ・単一分岐構造 if 文
  - ・多重分岐構造 else-if 文
  - ・ケース構造 switch 文
- 反復構造
  - ・前判定型反復構造 while 文, for 文
  - ・後判定型反復構造 do-while 文

4.1 if文

```
if（条件式）
    処理;
```

分岐の条件が１つだけの場合を単一分岐という.（図:1 参照）

sample3.c

```
/*if 文の例題*/
#include <stdio.h>
void main(){
int n;
scanf("%d", &n);
if ( 10 <= n && n < 20 ){
   printf("10 代\n");
   }
}
```

4.2 if-else文

```
if（条件 1）{
   処理 1;
}
else if（条件 2）{
    処理 2;
}
     |
     |
else if（条件 n）{
     処理 n;
}
else {
     処理;
}
```

多重分岐とは，単一分岐を組み合わせたもので，単一分岐の処理の中にまた条件分岐を含むものである．分岐の条件が１つだけの場合を単一分岐という.（図:1 参照）

sample4.c

```
/*else-if 文の例題*/
#include <stdio.h>
void main(){
    int  n;
    scanf("%d",&n);
    if ( n < 0 ){
       printf("負数\n");
    }else{
       if ( n == 0 ){
           printf("0\n");
        }else{
           printf("正数\n");
        }
     }
}
```

4.3 switch文

```
switch（式）{
case 定数式 1:
        処理 1;
        break;
    case 定数式 2:
        処理 2;
        break;
     |
     |
    case 定数式 n:
       処理 n;
        break;
    default:
       処理;
        break;
}
```

３つ以上の異なった処理の中から，条件にあった処理を行うことを多方向分岐と言い，次のような書き方をする．（図:1 参照）

sample5.c

```
/*switch 文の例題*/
#include <stdio.h>
void main(){
    int w;
    char  a;
    scanf("%c",&a);

    switch(a){
      case 'A': w=65;
                break;
      case 'B': w=66;
                break;
      default:  w=0 ;
    }
   printf("w=%d\n",w);
}
```

4.4 while 文

```
while（継続条件式）{
    処理;
}
```

条件式を前判定して反復制御を行う．継続条件が真である間，処理を繰り返し実行．継続条件式が始めから偽の場合は 一度も実行されない．（図:1 参照）

sample6.c

```
/*while 文の例題*/
#include <stdio.h>
void main(){
    int i;
    int a[5]={100,200,300,400,-1};
    i =0;
    while ( a[i] != -1) {
       printf("%d\n",a[i]);
       i++;
    }
}
```

4.5 do-while 文

```
do  {
    処理;
} while（継続条件式）;
```

まず処理を実行してから，継続条件の判定を行う．継続条件式が真である間，文を繰り返し実行．while 文は一度も実行されないことがあるが（最初から条件が’偽’のとき），do〜while 文ではとりあえず 1 回は文を実行する．（図:1 参照）

sample6.c

```
/*do-while 文の例題*/
#include <stdio.h>
int main(void)
{
    int sum, data;
    sum = 0;
    do {
        printf("整数値を入力");
        scanf("%d", &data);
        sum = sum + data;
        printf("sum = %d\n", sum);
    }while(data != 0);

    return 0;
}
```

4.6 for 文

```
for（初期化式; 継続条件式; 再初期化式）{
    処理;
}
```

定められた回数だけ反復制御を行う．（図:1 参照）

sample7.c

```
/*for 文の例題*/
#include <stdio.h>
int main(void)
{
int i;
int sum = 0;
long mul = 1;

for(i = 0; i < 9; i++) {
    sum = sum + i;
    mul = mul * i;
}
```

5 関数

C 言語の基本構成単位は関数である．関数の組み合わせによってプログラムが作られている．C 言語には「printf」に代表されるように標準関数という関数が用意されている．しかし標準関数だけでなくユーザーが関数を作ることもできる．今までのプログラムでは自分で作った関数は「main」関数だけです．これは処理が単純だったため，main 関数 1 つだけで何とかなっていた．(C 言語では main 関数が一番最初に実行されるという決まりがある) しかしプログラムが大きくなり，処理が複雑になると関数をいくつか作る必要がある．

自分で作る関数は機能単位にすべきある．これは，プログラムの再利用性を高めるためである．例えば，表示する関数「printf」は表示する機能をまとめることによって，いろいろなプログラムで使うことができる．このように，関数を作るときは，他のプログラムで使用するときも何も変更しないで良いように，ある一つの機能を持たせ，そのプログラム専用にならないように設計すべきである．

関数を作る前に「main」関数を見直してみると，

```
void main()
{
    int a;
    int b;
      :
      :
}
```

「main」というのは関数の名前．その前の「void」というのは関数の型．関数も値を持ちます．その関数がどのような型の値を持つかを表す．void というのは，英語では,「空間，空洞」などの意味を持ちます．つまり，値を持たないと言う型．

「int　a；」というのは，その関数の中で使う変数の宣言です．このようにある関数の中でしか使わない変数を「ローカル変数 」と言う．ローカル変数は，他の関数内のローカル変数と同じ名前があっても構わない．(もちろん同じ関数内のローカル変数は名前が同じではダメ！) また，ローカル変数が必要なければ省略することもできる．書式は下記のようになる．

```
[型] 関数名 ([引数宣言]){
     [ローカル変数の宣言]
    処理

    [return 値]
}
```

[型] は関数の型．省略すると「int」型になりますが，きちんと宣言するべきです．関数名は関数の名前．前述した規則を守れば，自由に決めることができる．ただし，同じ名前はダメです．(C++ 言語では，同じ名前があっても引数が異なれ命名できる！) 関数名は，名前を見ればどんな処理をする関数かすぐわかるように命名するべきである．

[引数宣言] は引数の宣言で，必要な個数記述する．省略すると引数はないという意味．

[ローカル変数の宣言] も必要なければ省略可能．

処理は，関数本体です．省略しても良いですが，関数の意味が無くなります．

[return 値] は関数の型が void 型の場合は省略できる．return の後が関数の値になる．式の型と関数の型は同一でなければならない．

sample8.c

```
/*単純な関数*/
#include <stdio.h>

void fun();

void main(){
    fun();
}
```

sample9.c

```
/*返り値ありの関数*/
#include <stdio.h>

int plusint(int a, int b){
    return a + b;
}

void main(){
    int  i1, i2;
    int  total;

    scanf("%d %d", &i1, &i2);
    total = plusint(i1, i2);

    printf("%d\n", total);
}
```

6 練習問題

1. n個の実数を入力し，和，積，平均，最大値を計算するプログラムを作成せよ．
2. キーボードから 16 進数を１つ入力し，それを 10 進数で表示するプログラムを作成せよ．
3. キーボードから月 (1〜12) を入力し，その月の日数を表示するプログラムを作成せよ．
4. 次の２つのプログラムの実行結果を答えよ．また単項演算子を使用せず，複合代入演算子を用いて書き換えよ．

```
#include <stdio.h>

void main(){
int  i = 0;
while(i++<10)
printf("%d\n", i);
}
```

```
#include <stdio.h>

void main(){
int  i = 0;
while(++i<10)
printf("%d\n", i);
}
```

7 おまけ

C 言語に触れる上で参考になる本，サイトです．また google もかなり使えます．google さんとはお友達になりましょう．

参考文献

[1] ハーバート シルト (著), Herbert Schildt (原著), トップスタジオ (翻訳), 柏原 正三: ”独習 C”, 翔泳社 ; ISBN: 4798102962

[2] 椋田實 : ”改訂第 4 版 ANSI C 対応 はじめての C”，ISBN4-7741-1264-X

[3] http://www5c.biglobe.ne.jp/ ecb/c/c00.html

[4] http://www9.plala.or.jp/sgwr-t/

[5] http://www.sgnet.co.jp/c/

表 1: データ型

データの種類	型	バイト長	数値の範囲
整数	(signed) char	1	-128〜+127
整数	unsigned char	1	0〜+255
整数	(signed) short (int)	2	-32768〜+32767
整数	unsigned short (int)	2	0〜+65535
整数	(signed) long (int)	4	-2147483648〜+2147483647
整数	unsigned long (int)	4	0〜+4294967295
整数	(signed) int	2 or 4	(signed) short (int) or (signed) long (int)
整数	unsigned int	2 or 4	(signed) short (int) or (signed) long (int)
小数	float	4	仮数部 2^{32}, 指数部 2^{8}
小数	double	8	仮数部 2^{52}, 指数部 2^{11}

表 2: 変換指定文字列

型指定文字	意味	使われるデータ型
%c	1 文字として出力	char 型
%d	10 進数で出力	int 型
%x	16 進数で出力	int 型
%o	8 進数で出力	int 型
%ld	long 型変数を 10 進数で出力	long 型
%f	[-]dddd.dddddd の形式で出力	float 型 double 型
%e	指数形式で出力	float 型 double 型
%s	文字列として出力	char 型配列

表 3: フラグ

-	左詰めに表示（省略時には右詰め）
+	符号を付ける（省略時には ”-” 符号のみ）
#	数値の表記形式がわかるように表示 %#o のとき：数字の前に ”0” を付加 %#x のとき：数字の前に ”0x” を付加
0	0 を詰める

表 4: 書式指定文字列

型指定文字	意味	使われるデータ型
%c	1 文字として入力	char 型
%d	10 進数で入力	int 型
%x	16 進数で入力	int 型
%o	8 進数で入力	int 型
%ld	long 型変数を 10 進数で入力	long 型
%f	float 型変数に実数を入力	float 型
%lf	double 型変数に実数を入力	double 型
%s	文字列として入力	char 型配列

表 5: エスケープシーケンス

エスケープシーケンス	意味	ASCII コード（16 進）
¥n	復帰改行	0A
¥a	警報音	07
¥t	タブコード	09
¥b	バックスペース	08
¥¥	文字としての ¥	5C
¥’	文字としての ’	2C
¥”	文字としての ”	22
¥0	文字列終了コード	0

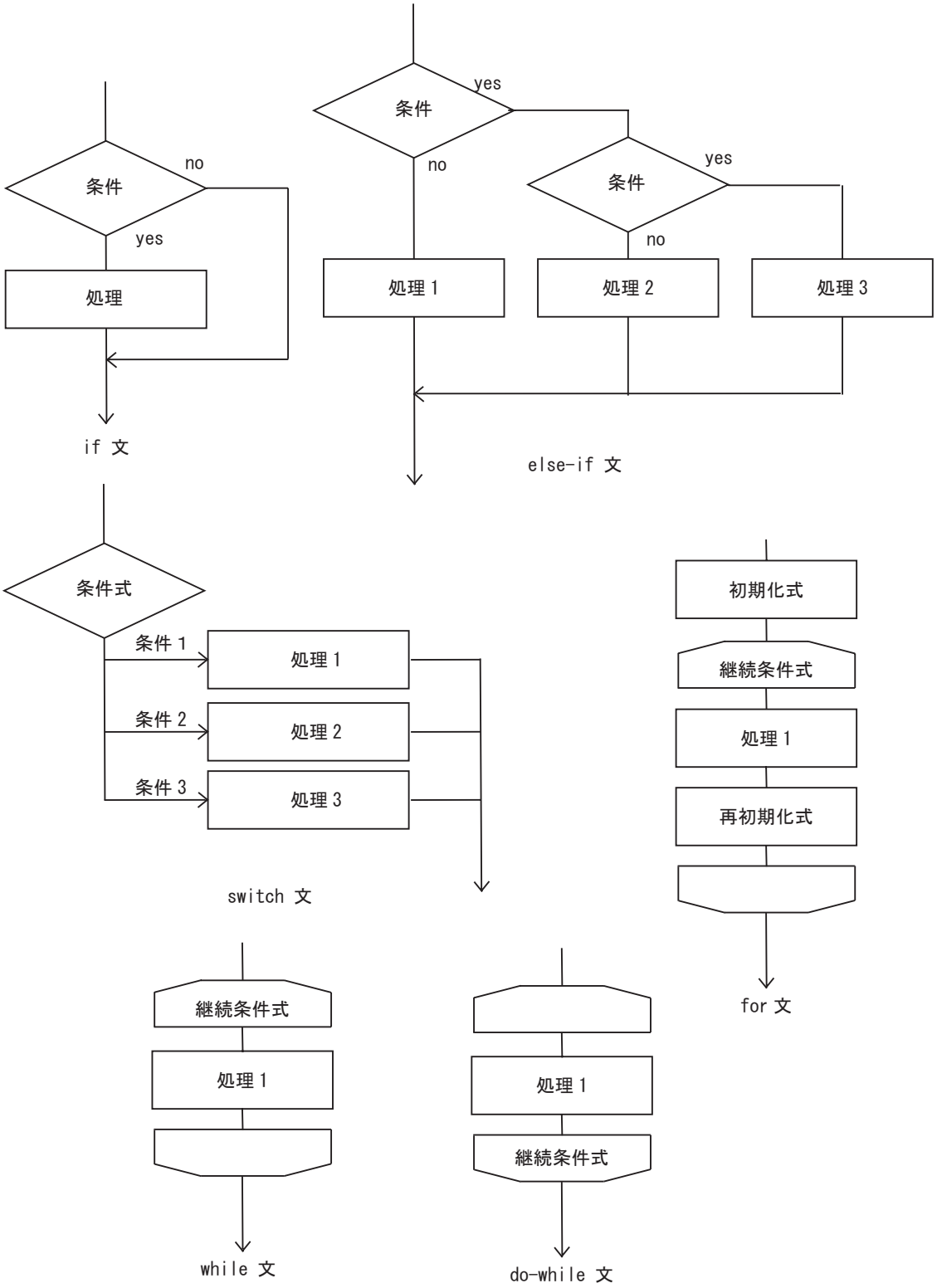
条件
no
yes
処理
if 文
条件
yes
no
処理 1
条件
yes
no
処理 2
処理 3
else-if 文
条件式
条件 1
処理 1
条件 2
処理 2
条件 3
処理 3
switch 文
初期化式
継続条件式
処理 1
再初期化式
for 文
継続条件式
処理 1
while 文
処理 1
継続条件式
do-while 文

図 1: フローチャート