3　データ型

3.1　整数

3.1.1　2 進数、8 進数、10 進数、16 進数

10 進数の 234 は、

$$234 = 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$$

と表わされることに基づいている。100 が 2 個、10 が 3 個、1 が 4 個あるので二百三十四と呼ばれる。

10 進数の 43 は、

$$43 = 4 \times 10^1 + 3 \times 10^0 = 2^5 + 2^3 + 2^1 + 2^0 = 5 \times 8^1 + 3 \times 8^0 = 2 \times 16^1 + 11 \times 16^0$$

と表わせるので、2 進数、8 進数、16 進数による表記は

$$43_{(10)} = 101011_{(2)} = 53_{(8)} = 2\mathrm{B}_{(16)}$$

となる[7]。

10 進数、2 進数、8 進数、16 進数の対応表を次に示す。

10 進	2 進	8 進	16 進	10 進	2 進	8 進	16 進	10 進	2 進	8 進	16 進
0	0	0	0	8	1000	10	8	16	10000	20	10
1	1	1	1	9	1001	11	9	17	10001	21	11
2	10	2	2	10	1010	12	A	18	10010	22	12
3	11	3	3	11	1011	13	B	19	10011	23	13
4	100	4	4	12	1100	14	C	20	10100	24	14
5	101	5	5	13	1101	15	D	21	10101	25	15
6	110	6	6	14	1110	16	E	22	10110	26	16
7	111	7	7	15	1111	17	F	23	10111	27	17

2 進数を固定桁数で表すとき、固定桁数が表示する数の桁数より大きい場合は、先頭を 0 で埋める。例えば 10 進数の 5 を 8 桁の 2 進数で表すと 00000101 となる。固定桁数が表示する数の桁数より小さい場合は、固定桁数を超えた部分は無視する。例えば 10 進数の 261 を 2 進数で表すと 100000101 となるので、8 桁の 2 進数で表すと (10 進数の 5 と同じ) 00000101 となる。

2 進数を 4 桁 (3 桁) ごとに下の桁から上の表に基づいて 16 進 (8 進) に直すと、16 進数 (8 進数) が得られる。逆に、16 進数 (8 進数) を 2 進数に変換するときは、各桁を上の表に基づいて 2 進に直す。8 進数と 16 進数の変換は、2 進数を仲立ちにして行う。

3.1.1.1　10 進数を 16 進数へ

10 進数を 16 進数に変換するには、商を次々に 16 で割り商が 0 になったとき余りを下の桁から並べる。$2010_{(10)}$ を例に取ると

$$\begin{array}{r|r|l} 16 & 2010 & 10{=}\mathrm{A}_{(16)} \\ \hline 16 & 125 & 13{=}\mathrm{D}_{(16)} \\ \hline 16 & 7 & 7 \\ \hline & 0 & \end{array}$$

[7] N を自然数 ($N \geqq 2$) とする。自然数 m を

$$m = c_0 N^n + c_1 N^{n-1} + \ldots + c_n, \quad c_0 \neq 0, \quad 0 \leqq c_j < N \ (j = 1, \ldots, n)$$

と表わしたとき、

$$c_0 c_1 \ldots c_{n(N)}$$

を m の N 進表現という。N を基数という。基数 N が文脈から明らかなときは添え字 (N) は省略する。我々が日常用いている数は $N = 10$ の 10 進数であるが、計算機科学では、$N = 2,\ 8,\ 16$ がよく用いられる。$N = 8$ のときは 8 進数 (octal number) といい、0 から 7 までの文字を用いて表わす。$N = 16$ のときは 16 進数 (hexadecimal number) といい、0 から 9 までの数字と A から F までの英字 (または a,...,f) で表わす。

だから、

$$2010_{(10)} = 7\mathrm{DA}_{(16)}$$

となる。これは

$$2010 = 16 \times 125 + 10 = 16 \times (16 \times 7 + 13) + 10 = 7 \times 16^2 + 13 \times 16 + 10$$

より分かる。

3.1.1.2　10 進数を 2 進数へ　10 進数を 2 進数に変換するには、商を次々に 2 で割り商が 0 になったとき、余りを下の桁から並べる。$2010_{(10)}$ を例に取ると

2	2010	0
2	1005	1
2	502	0
2	251	1
2	125	1
2	62	0
2	31	1
2	15	1
2	7	1
2	3	1
2	1	1
	0	

より、余りを下から並べて

$$2010_{(10)} = 111\ 1101\ 1010_{(2)}$$

となる。分かりやすいように下から 4 桁毎に空白を入れた。

10 進数を 16 進数に変換し、次いで 16 進数を 4 桁の 2 進数に変換しても得られる。16 進数の最高位以外の桁が $7_{(16)}$ 以下のときは、4 桁の 2 進数になるよう先頭を 0 で埋める必要がある。たとえば、10 進整数の 2010 は

$$2010_{(10)} = 7\mathrm{DA}_{(16)} = 111\ 1101\ 1010_{(2)}$$

となる。

10 進数を 8 進数に変換するのも同様である。

演習 3.1 [第 2 種情報処理技術者試験 2000 年度春期午前]

すべて同じ値を表している 2 進数、8 進数、10 進数、16 進数の組合わせはどれか。

	2 進数	8 進数	10 進数	16 進数
ア	111	10	8	8
イ	1010	12	10	A
ウ	1100100	256	100	84
エ	1111111	377	256	FF

注 3.1 第 2 種情報処理技術者試験は情報処理技術者試験 (基本情報技術者) の前身である。

3.1.2 補数

1 桁の N 進整数 c と 10 進整数 $d > 0$ に対し、c_d は c が d 個続いた数とする。たとえば、2 進整数として $1_5 0_3 = 11111000$ である。

d 桁の N 進数 x に対し $10_d - x$ を x **の** N **の補数**、$10_d - 1 - x = 1_d - x$ を x **の** $N-1$ **の補数**という。補数を使うことにより、減算を加算で行うことができるので、整数をコンピュータ内部で表現する際に用いられる。

例 3.1 3 桁の 10 進数を考える。123 に対し、10 の補数は 877(= 1000 − 123) であり、9 の補数は 876(= 999 − 123) である。−123 に対し、10 の補数は 123(= 1000 − (−123) の下 3 桁) であり、9 の補数は 122(= 999 − (−123) の下 3 桁) である。356 − 123 を 3 桁の 10 進数で 10 の補数を用いて計算すると、

$$356 - 123 \equiv 356 + 877 = 1233 \equiv 233$$

となる。ここで、≡ は 3 桁の固定桁数表示で等しいことを表す。

3.1.2.1 2 の補数

d 桁 (ビット) の 2 進数 $b(> 0)$ を考える。$10_d - b$(1 の後 0 が d 個続く数から b を引く) が b の 2 の補数 (two's complement) である。0 の 2 の補数は 0 である。

8 ビットの 2 進数 $b(> 0)$ を考える。b の 2 の補数は、$100000000_{(2)} - b$ である。

b の 2 の補数は

1. すべてのビットを反転 (0 を 1、1 を 0)　(= 1 が d 個続く 2 進数から b を引く)
2. 1 を加える　(= 1 の後 0 が d 個続く 2 進数から b を引く)

により得られる。

例 3.2 8 ビットの 2 進数 00110101 の 2 の補数は 11001011 である。

1	1	0	0	1	0	1	0	すべてのビットを反転
1	1	0	0	1	0	1	1	1 を加える

3.1.2.2 1 の補数

b を d 桁 (ビット) の 2 進数とする。$1_d - b$(1 が d 個続く数から b を引く) が b の 1 の補数 (one's complement) である。

8 ビットの 2 進数 $b(> 0)$ を考える。b の 1 の補数は、$11111111 - b$ である。b の 1 の補数はすべてのビットを反転 (0 を 1、1 を 0) して得られる。

例 3.3 8 ビットの 2 進数 00110101 の 1 の補数は 11001010 である。

演習 3.2 [情報処理技術者試験 (基本情報技術者)2002 年度春期午前]

4 ビットの 2 進数 1010 の 1 の補数と 2 の補数の組合わせはどれか。

	1 の補数	2 の補数
ア	0101	0110
イ	0101	1001
ウ	1010	0110
エ	1010	1001

注 3.2 2006 年度以降の情報処理技術者試験の問題冊子・配点割合・解答例・採点講評は主催者の独立行政法人情報処理推進機構が Web

`http://www.jitec.jp/`

で公開している。

情報処理技術者試験 (基本情報技術者) および第 2 種情報処理技術者試験の問題と解答例は
東京理科大学情報処理技術者試験研究会 http://www.rs.kagu.tus.ac.jp/infoserv/j-siken/
で見ることができる。
2002 年度以降の情報処理技術者試験 (基本情報技術者) の問題と解答例と解説は
http://情報処理試験.jp/
で見ることができる。

3.1.3 ASCII(アスキー)

コンピュータでは、文字は 0 と 1 の並びで表される。0 と 1 の並びを 2 進数による数と考えると、文字と数の対応を決めることが必要になる。文字の集まりを文字集合といい、数との対応まで決めたものが符号化文字集合 (coded character set) である。

1963 年アメリカ標準協会 (American Standard Association) が 4、6、7 ビットの符号の標準化を行った。1968 年に 7 ビットに一本化されたのが ASCII 符号 (American Standard Code for Information Interchange 情報交換用米国標準符号) である。1967 年には ISO 符号となり、1969 年には JIS 符号にも採り入れられた。

ターミナルで man ascii とするとアスキーコードが 8 進、10 進、16 進で表示される。(終了は q)

```
$ man ascii
(中略)
     The hexadecimal set:

     00 nul   01 soh   02 stx   03 etx   04 eot   05 enq   06 ack   07 bel
     08 bs    09 ht    0a nl    0b vt    0c np    0d cr    0e so    0f si
     10 dle   11 dc1   12 dc2   13 dc3   14 dc4   15 nak   16 syn   17 etb
     18 can   19 em    1a sub   1b esc   1c fs    1d gs    1e rs    1f us
     20 sp    21  !    22  "    23  #    24  $    25  %    26  &    27  '
     28  (    29  )    2a  *    2b  +    2c  ,    2d  -    2e  .    2f  /
     30  0    31  1    32  2    33  3    34  4    35  5    36  6    37  7
     38  8    39  9    3a  :    3b  ;    3c  <    3d  =    3e  >    3f  ?
     40  @    41  A    42  B    43  C    44  D    45  E    46  F    47  G
     48  H    49  I    4a  J    4b  K    4c  L    4d  M    4e  N    4f  O
     50  P    51  Q    52  R    53  S    54  T    55  U    56  V    57  W
     58  X    59  Y    5a  Z    5b  [    5c  \    5d  ]    5e  ^    5f  _
     60  `    61  a    62  b    63  c    64  d    65  e    66  f    67  g
     68  h    69  i    6a  j    6b  k    6c  l    6d  m    6e  n    6f  o
     70  p    71  q    72  r    73  s    74  t    75  u    76  v    77  w
     78  x    79  y    7a  z    7b  {    7c  |    7d  }    7e  ~    7f del
(後略)
```

アルファベット 2 文字ないし 3 文字で表わした $00_{(16)}$ から $20_{(16)}$ までと $7\mathrm{F}_{(16)}$ は、制御文字である。$21_{(16)} \sim 7E_{(16)}$ は図形文字という。

3.2 整数型

オブジェクトはメモリ上に置かれる。どれだけ (何ビット) のメモリをどのように使うかによって、いろいろな型に分かれる。型には文字を扱う文字型、整数を扱う整数型、実数や複素数を扱う浮動小数点型がある。本節では文字型と整数型を扱う。

JIS(X3010:2003) で「処理系定義」「その値以上の大きさを持たねばならない」などとされている場合、東女の現システムの値を「東女」として示してある。

3.2.1 char 型

図形文字、空白文字、いくつかの制御文字 (逆斜線\と英字で表す)、を格納する型である。東女では大きさは 8 ビット (1 バイト) で、値はアスキーコードの値に一致する。

例 3.4 アルファベットの小文字 a (アスキーコード $61_{(16)} = 01100001_{(2)}$) を char 型のオブジェクトに代入すると、東女ではコンピュータのメモリに

0	1	1	0	0	0	0	1

と格納される。

制御文字の逆斜線表記 (escape sequence)

文字	2 進数 (東女)	16 進数 (東女)	意味	動作
\0	00000000	00	ナル文字 (null character)	(文字列の終りを示す。)
\a	00000111	07	警報 (alert)	視覚的または聴覚的警報を発する。
\b	00001000	08	後退 (backspace)	その行の直前の位置に移動。
\t	00001001	09	水平タブ (horizontal tab)	その行の水平タブの位置に移動
\n	00001010	0a	改行 (new line)	次の行の最初の位置に移動。
\v	00001011	0b	垂直タブ (vertical tab)	垂直タブの行の最初の位置に移動。
\f	00001100	0c	書式送り (form feed)	次の頁の最初の位置に移動。
\r	00001101	0d	復帰 (carrige return)	その行の最初の位置に移動。

3.2.2 文字型

文字型 (character type) には、型 char、負の数を扱わない符号なし char 型 (unsigned char type) と負の数を扱う符号付き char 型 (signed char type) がある。

符号付き char 型では、最上位ビットが 1 のとき負の数とする。表し方は、

1. 符号と絶対値 (例 10000001 は $-1_{(10)}$)
2. 2 の補数 (例 10000001 は $-127_{(10)}$)
3. 1 の補数 (例 10000001 は $-126_{(10)}$)

の 3 通りがある。

符号なし char 型 (unsigned char 型)

メモリ	10 進数
00000000	0
00000001	1
00000010	2
...	
01111111	127
10000000	128
10000001	129
...	
11111111	255

符号付き char 型 (signed char 型)

メモリ	2 の補数	1 の補数
	10 進整数	
00000000	0	0
00000001	1	1
00000010	2	2
...		
01111111	127	127
10000000	−128	−127
10000001	−127	−126
...		
11111111	−1	−0

型 char の扱いはシステムにより異なる (JIS では「結果の値は処理系定義」と表現している) が東女では signed char 型で負数は 2 の補数を用いている。次のプログラムにより確認することができる。

```
1   /* char 型の種類 */
2   /* chartype.c */
3   #include <stdio.h>
4
5   int main(void)
6   {
7     char c;
8
9     c='\xff';
10    printf("c=%d \n",c);
11
12    return 0;
13  }
```

7 行目で char 型オブジェクト c を宣言し、9 行目で c に文字型定数'\xff' を代入している。(文字定数については次節で述べる。) c のすべてのビットが 1 になる。10 行目の printf() で、255、-127、-1、0 のどれが出力されるかにより、char 型の種類、負の数の表示法が分かる。

printf() の出力	char 型の種類	負の数
255	unsigned char	ない
-127	signed char	符号と絶対値
-1		2 の補数
0		1 の補数

演習 3.3 [基本情報処理技術者試験 2006 年度秋期午前]
1 バイトのデータで 0 のビット数と 1 のビット数が等しいもののうち，符号なしの 2 進整数として見たときに最大になるものを，10 進整数として表したものはどれか。

ア 120　　イ 127　　ウ 170　　エ 240

演習 3.4 [基本情報処理技術者試験 2006 年度秋期午前]
負数を 2 の補数で表す 16 ビットの符号付き固定小数点数の最小値を表すビット列を, 16 進数として表したものはどれか。

ア 7FFF　　イ 8000　　ウ 8001　　エ FFFF

printf() 関数で出力する際、%に続いて変換指定子 (conversion specifier) を用いて文字かあるいは何進数かを区別して表示することができる。

種類	変換指定子
文字	c
10 進整数	d
10 進整数 (符号なし)	u
8 進整数	o(小文字のオー)
16 進整数	x, X

注 3.3 2 進整数で表す変換指定子は用意されてない。

変換指定子による出力の違いは次のプログラムにより確認することができる。

```
1  /* char.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6    char c;
7
8    printf("1 文字入力して下さい。");
9    scanf("%c",&c);
10   printf(" 文字 %c",c);
11   printf("10 進整数 %d",c);
12   printf(" 8 進整数 %o",c);
13   printf("16 進整数 %x",c);
14
15   return 0;
16 }
```

実行結果は、

```
% cc -o char char.c -Wall
% ./char
1 文字入力して下さい。a
    文字 a
10 進整数 97
 8 進整数 141
16 進整数 61
```

演習 3.5 次のように画面に出力するプログラム `today.c` を作れ。

```
今日は何月ですか？
5 [return]
今日は何日ですか？
21 [return]
今日は 5 月 21 日です。
```

演習 3.6 キーボードから (21 から 7E までの)16 進数を入力し、アスキーコードが入力した数に一致する記号を出力するプログラム hex2char.c をつくれ。(たとえば、61 と入力すると a が出力される。) 16 進数で入力する変換指定子は x である。

3.2.3 整数型の大きさによる分類

整数型データは何ビットで表すか、符号を付けるか否かにより次表のように分かれる。符号つき整数型の負の数を東女では 2 の補数を用いて表す。

データ型		大きさ (東女)	範囲 (東女)
`char`	文字型	1 バイト	$-128 \sim 127$
`signed char` †	符号付き整数型	1 バイト	$-128 \sim 127$
`signed short`		2 バイト	$-32768 \sim 32767$
`int`		4 バイト	$-2147483648 \sim 2147483647$
`long int`		4 バイト	$-2147483648 \sim 2147483647$
`long long int`		8 バイト	$-9223372036854775808 \sim 9223372036854775807$
`unsigned char` ‡	符号なし整数型	1 バイト	$0 \sim 255$
`unsigned short`		2 バイト	$0 \sim 65535$
`unsigned int`		4 バイト	$0 \sim 4294967295$
`unsigned long int`		4 バイト	$0 \sim 4294967295$
`unsigned long long int`		8 バイト	$0 \sim 18446744073709551615$

† 文字型ともいえるし符号付き整数型ともいえる。
‡ 文字型ともいえるし符号なし整数型ともいえる。

3.2.3.1　長さ修飾子　変換指定子は適用する変換の種類を指定するのに対し、実引数の大きさを指定する長さ修飾子 (length modifier) がある。長さ修飾子は省略可能である。次表の長さ修飾子は整数型の変換指定子 (d, i, o, u, x, X) の長さを指定する。

長さ修飾子	対応する実引数の型
l(小文字のエル)	long int, または unsigned long int
ll(小文字のエル 2 つ)	long long int, または unsigned long long int

3.2.3.2　フラグ　変換指定子には長さ修飾子以外に出力のフィールド幅 (桁数) や右詰めか左詰めかなどを指定するフラグをつけることもできる。-フラグを指定すると左詰めで、指定しないと右詰めで出力される。%(フラグが指定されているときはフラグ) の後に正の整数 n を指定するとつけるとフィールドの大きさが n となる。

例 3.5　1. printf("%5d\n",12);

			1	2

2. printf("%-5d\n",12);

1	2			

3. printf("%15lld\n",123456789012LL); (LL は long long int 型の定数を指定する接尾語である。)

			1	2	3	4	5	6	7	8	9	0	1	2

演習 3.7 例 3.5 をプログラムをつくり確かめよ。3 において、LL をつけないでコンパイルするとどうなるか。

3.2.3.3　size_t 型と sizeof 演算子　sizeof 演算子の結果はオペランドのバイト数である。オペランドは式 (オブジェクトは式である) または型の名前を括弧 () で囲ったものである。結果の値はシステムによって異なり、型は size_t 型である。size_t 型は東女では unsigned long int 型である。

符号付き整数型の大きさと範囲は次のプログラムにより確認することができる。

```
1   /* 整数型の大きさと範囲 */
2   /* integer.c */
3   #include <stdio.h>
4   #include <limits.h>
5
6   int main(void)
7   {
8     printf(" char:%3lu byte %d〜%d\n", sizeof(char),CHAR_MIN,CHAR_MAX);
9     printf(" short:%3lu bytes %d〜%d\n", sizeof(short),SHRT_MIN,SHRT_MAX);
10    printf(" int:%3lu bytes %d〜%d\n", sizeof(int),INT_MIN,INT_MAX);
11    printf(" long:%3lu bytes %ld〜%ld\n", sizeof(long),LONG_MIN,LONG_MAX);
12    printf("long long:%3lu bytes %lld〜%lld\n", sizeof(long),LLONG_MIN,LLONG_MAX);
13
14    return 0;
15  }
```

4 行目は ヘッダ <limits.h> を取り込んでいる。<limits.h> には整数型データの大きさなどが記述されている。

8 行目から 11 行目の CHAR_MIN、SHRT_MIN、INT_MIN、LONG_MIN および LONG_MIN は、それぞれ char 型、short 型、int 型、long 型の最小数を規定する定数で、limits.h ファイルに記述されている。CHAR_MAX、SHRT_MAX、INT_MAX、LONG_MAX はそれぞれ最大数である。

8 行目から 11 行目の sizeof は、オペランドの型あるいはオブジェクトが何バイトの大きさかを与える演算子である。東女では符号なし long int 型であるが、Cygwin では符号なし int 型である。

演習 3.8 26 ページの `integer.c` に類似のプログラム `unsigned.c` を作り、符号なし整数型について確かめよ。ただし、unsigned char, unsigned short, unsigned int, unsigned long int, unsigned long long int の最大数は、<limits.h> にそれぞれ `UCHAR_MAX,USHRT_MAX,UINT_MAX,ULONG_MAX,ULLONG_MAX` として与えられている。最小値はいずれの型も 0 である。(したがって、`UCHAR_MIN,USHRT_MIN,UINT_MIN,ULONG_MIN,ULLONG_MIN` は定義されてない。) 符号なし整数型、unsigned long int 型、unsigned long long int 型の (長さ修飾子と) 変換指定子はそれぞれ `u, lu, llu` である。

3.3 浮動小数点型

3.3.1 固定小数点と浮動小数点

コンピュータでは実数を有限桁で近似する。表示方法には固定小数点表示と浮動小数点表示がある。

固定小数点数 (fixed point number)
整数部分に用いる桁数と小数部分に用いる桁数をあらかじめ固定して表現した数である。整数は小数部分に用いる桁数を 0 とした固定小数点数と考えることができる。

浮動小数点数 (floating point number)
数を (符号部)(仮数部) $\times$ (基数)$^{(\text{指数部})}$ として表現した数である。符号部は $+$ または $-$ で、$+$ の場合は省略されることがある。仮数部 (mantissa) は有効数字部 (significant part) とも呼ばれる。指数部 (exponent part) をかえることにより小数点の位置が動くので浮動小数点数と呼ばれる。6.0221415×10^{23} の場合、6.0221415 が仮数部、10 が基数 (base/radix)、10^{23} が指数部である。

例 3.6 10 進小数 123.456 は固定小数点表示、浮動小数点表示の方法は何通りもある。$0.0123456 \times 10^4 = 0.123456 \times 10^3 = 1.23456 \times 10^2 = 12.3456 \times 10^1 = 123.456 \times 10^0 = \cdots$

正の数に対し、仮数部を 1 以上基数未満とする表し方を正規化という。本例の場合、1.23456×10^2 が正規化で、仮数部は 1.23456、指数部は 10^2 である。10 は基数、2 は指数である。

3.3.2 実浮動小数点型

C では、(整数でない) 実数は実浮動小数点型 (real floating type) として表す。実浮動小数点型には float 型、double 型および long double 型がある。float、double、long double の大きさと範囲を次表に示す。

データ型	大きさ (東女)	範囲 (東女)
`float`	4 バイト	約 $1.2 \times 10^{-38} \sim 3.4 \times 10^{38}$
`double`	8 バイト	約 $2.2 \times 10^{-308} \sim 1.8 \times 10^{308}$
`long double`	8 バイト	約 $2.2 \times 10^{-308} \sim 1.8 \times 10^{308}$
`long double`	16 バイト	約 $3.4 \times 10^{-4932} \sim 1.2 \times 10^{4932}$

注 3.4 long double を含むソースをコンパイルすると

```
warning: use of 'long double' type; its size may change in a future release
```

との警告が表示されるシステムもある。

3.3.2.1 変換指定子と長さ修飾子 1.2345×10^{-2} を 1.2345e-2(1.2345E-2) と表わすのが e 変換 (E 変換)、0.012345 と表わすのが f 変換である。e 変換 (E 変換) と f 変換の適切な方[8]で表示するのが g 変換 (G 変換) である。a 変換 (A 変換) は仮数部を 16 進小数、指数部は 2 のべき乗を 10 進整数で表す。

float 型、double 型、long double 型の長さ修飾子と変換指定子を次表に示す。

[8] 指数部が -4 より小さいか、精度以上のときは e 変換、そうでないときは f 変換を用いる。

型	入力	出力
float	f	a, A, e, E, f, g, G
double	lf	a, A, e, E, f, g, G
long double	Lf	La, LA, Le, LE, Lf, Lg, LG

演習 3.9 double型データをキーボードから入力し、e変換、f変換、g変換で出力するプログラム double-io.c を作れ。e変換、f変換でいろいろな数 (0.0012345678、0.000012345678、123456789.0) を入力し試してみよ。

3.3.2.2 フラグ %と変換指定子の間に m.n とおくと最小フィールド幅は m で、精度は n 桁で出力される。精度はe、E、f変換のときは小数点文字の後に出力すべき桁数、g、G変換のときは最大の有効桁数を表す。フラグ-は結果をフィールド内に左詰めで出力する。(このフラグを指定しないと右詰めになる。) フラグ+は常に結果を正符号または負符号であじめる。(このフラグを指定しないと負の場合のみ符号をつける。)

例 3.7 1.2345×10^{-2} と 1.2345×10^{-256} を%10.3e、%10.3f、%10.3g で出力した結果を次表に示す。

1.2345×10^{-2}	%10.3e		1	.	2	3	5	e	-	0	2
	%10.3f						0	.	0	1	2
	%10.3g					0	.	0	1	2	3
1.2345×10^{-256}	%10.3e	1	.	2	3	5	e	-	2	5	6
	%10.3f						0	.	0	0	0
	%10.3g		1	.	2	3	e	-	2	5	6

演習 3.10 上記の例を確かめよ。フィールド幅の前に - フラグをつけるとどうなるか。

3.3.3 複素数型

C99 では、複素数の浮動小数点数を表す複素数型 (complex type) が導入された。複素数型を扱うには<complex.h>を取り込む必要がある。サンプルプログラムを示す。

```
1  /* complex.c */
2  #include <stdio.h>
3  #include <complex.h>
4
5  int main(void)
6  {
7    double x,y;
8    double complex z,w;
9
10   printf("実部を入力してください");
11   scanf("%lf",&x);
12   printf("虚部を入力してください");
13   scanf("%lf",&y);
14
15   z=x+y*I; /* I*I == -1 */
16   w=z*z;
17
18   printf("     z = %f %+fi\n", creal(z), cimag(z));
19   printf("carg(z) = %f\n", carg(z)); /* 偏角 */
20   printf("cabs(z) = %f\n", cabs(z)); /* 絶対値 */
21   printf("  z*z = %f %+fi\n", creal(w), cimag(w));
22
23   return 0;
24 }
```

複素数 z の実部と虚部をキーボードから入力し (10-13 行)、z および z の偏角 (単位はラジアン)、絶対値、平方を出力している (18-21 行)。

実浮動小数点型と複素数型を総称して浮動小数点型 (floating type) という。本講義では複素数型はこれ以上扱わない。