5 データ型と演算

5.1 文字と整数

5.1.1 C における文字の扱い方

- C のソースコードでは文字 a を 'a' と書く[*8][*9]
- 文字を格納する変数は，次のように宣言するのが一般的である。

  char 変数名;
- printf を用いた文字の表示例

```
char c = 'a';        /* 変数 c の宣言と初期化 */
printf("%c", c);     /* a と表示 */
printf("%c", 'c');   /* c と表示 */
```

- 注意: 'a' (文字 a) と "a" (文字列 a) は異なるものである

```
printf("a");         /* a と表示 */
printf('a');         /* 間違い */
```

- 文字を表示する関数（命令） putchar

```
char c = 'a';   /* 変数 c の宣言と初期化 */
putchar(c);     /* a と表示 */
putchar('c');   /* c と表示 */
putchar('\n'); /* 改行文字を表示（改行する） */
putchar("a");   /* 間違い */
```

5.1.2 コンピュータ内部での整数と文字の表現

- コンピュータ内部では，すべてのものは 0 と 1 の並びである
- 整数は 2 進数を使って 0 と 1 の並びとして表現される[*10]

  整数 1: 00000000 00000000 00000000 00000001

  整数 2: 00000000 00000000 00000000 00000010

  整数 3: 00000000 00000000 00000000 00000011
- 文字は ASCII (アスキー) コードに従い 0 と 1 の並びで表現される[*11]

  文字 'A': 01000001 (65)

  文字 'a': 01100001 (97)

  文字 '1': 00110001 (49)

  0 と 1 の並びとして表現された ASCII コード を 2 進数と見なせば，文字の内部表現を括弧内の 10 進整数で短く表記できる[*12]

[*8] 単に a と書けば変数 a の参照（内容の閲覧）である。

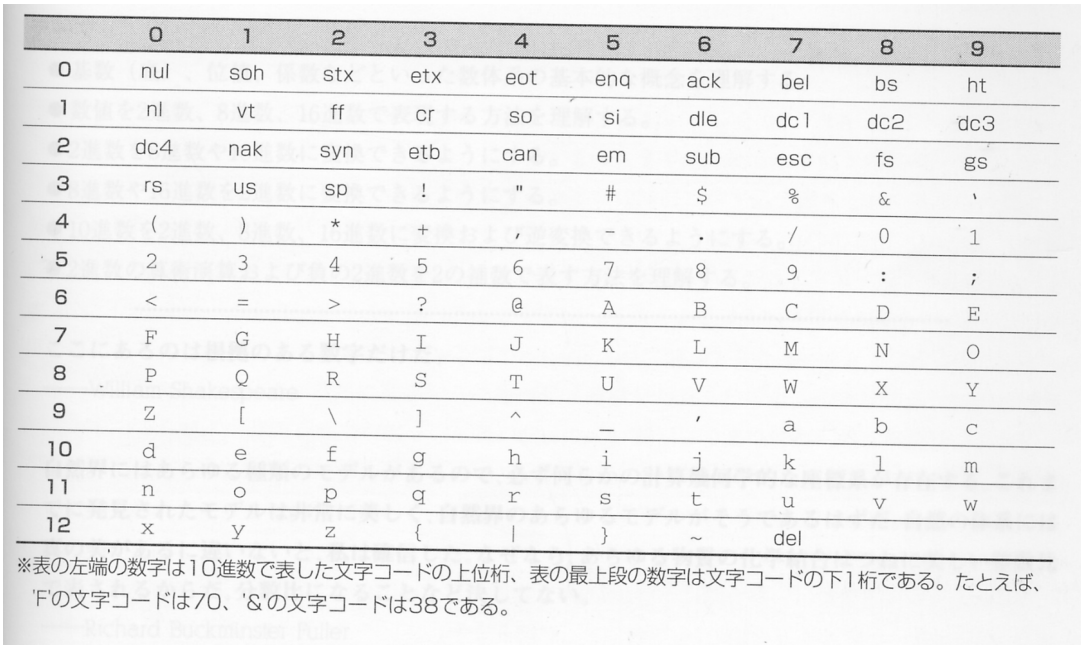
[*9] 'a' を文字定数 a と呼ぶ。

[*10] ただし，2 進数の各桁の並び順はコンピュータ (CPU) の種類によって異なる。

[*11] ここでは，いわゆる半角の英数字のみ考える。

[*12] ただし，ASCII コードは 16 進数で表記することの方が多い。

- ASCII コード（文字集合）



	0	1	2	3	4	5	6	7	8	9
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	nl	vt	ff	cr	so	si	dle	dc1	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	"	#	$	%	&	'
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	del		

※表の左端の数字は10進数で表した文字コードの上位桁、表の最上段の数字は文字コードの下1桁である。たとえば、'F'の文字コードは70、'&'の文字コードは38である。

- 注意：「数値（整数）の 1」と「数字（文字）の 1」は異なるものである
- printf の第一引数内の `%d` は，第 2 引数以降に与えられた値（実体は 0, 1 の並び）を 2 進数と見なして 10 進表示するためのもの
- printf の第一引数内の `%c` は，第 2 引数以降に与えられた値を文字として表示するためのもの

```
printf("%d,%d\n", 1, '1');  /* 1,49 と表示される */
printf("%c(%d)\n", 'a', 'a');  /* a(97) と表示される */
printf("%d\n", '1' + '1');  /* 98 と表示される */
```

5.1.3 データ型とサイズ

- コンピュータ内でのデータの表現は，すべて 0 と 1 の並びであるから，それを文字と見なすか，整数と見なすか，によってデータの意味するものが異なることになる。このようなデータの区別をデータ型 (type) という[*13]。
- 一つのデータを表すのに用いる 0 と 1 の個数はデータ型に応じて決まっている。この個数をデータ型のサイズという。
- これまでに登場したデータ型とサイズ
  - char 型: 8 bit (= 1 byte)
  - int 型: サイズは機種依存。最近のコンピュータでは 32 bit (= 4 byte) であることが多い。
- int 型で扱える整数の最小値と最大値は int 型のサイズによって決まる。
  int 型のサイズが 32 bit ならば，int 型で表現できる最小値は -2^{31} から $2^{31}-1$ [*14]

[*13] 1.0 などの実数は整数とは異なる方法で 0 と 1 の並びとして表現されている（浮動小数点方式）。
[*14] 2 の補数表現を使った 32 bit 整数の最小値と最大値

- sizeof 演算子を用いれば，各型のサイズを byte 単位で得ることができる[*15]。

■リスト 5.1　sizeof **演算子の使いかたの一例**

```
/* sizeof operator */
#include <stdio.h>

int main()
{
     printf("size of char: %d\n", sizeof(char));

     return 0;
}
```

実行結果

```
size of char:  1
```

5.1.4　文字を整数として扱う

- char 型のサイズは 8 bit であるから，char 型変数には 8 個の 0, 1 を格納できる。

0	0	0	0	0	0	0	0
⋮							
1	1	1	1	1	1	1	1

- char 型変数に格納された 0 1 の並びを整数と見なせば，char 型変数には 0 (2 進 00000000) から 255 (2 進 11111111) の範囲の整数を格納できると考えられる。
- int 型は文字を格納するのに十分なサイズを持つため，int 型変数に文字を代入することも可能。

```
char c;     /* char 型変数 c の宣言 */
int i;      /* int 型変数 i の宣言 */
c = 97;     /* 'a' を意味する 10 進整数値 97 を代入 */
printf("%c(%d)\n", c, c);  /* a(97) と表示される */
i = 'a';    /* i に 97 を代入 */
printf("%c(%d)\n", i, i);  /* a(97) と表示される */
```

文字 a の ASCII コードを 10 進で表現すれば 97 なので，`c = 'a'` と `c = 97` は同じこと。

5.1.5　キーボードからの数値と文字の入力

- キーボードからの入力はすべて文字である。

  例）キーボードから 12 とタイプすれば，整数の 12 ではなく，文字の '1' (00110001) と '2' (00110002) を続けて入力したことになる。

[*15] sizeof で変数等のサイズを調べることもできる。

- 例えば,

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

  によって int 型変数 i に整数を入力できるのは，キーボードから入力された数字の並びを scanf が 10 進数の整数と見なして int 型に変換するためである。

  例) キーボードからの入力 '1' '2' (00110001 00110002) は scanf によって 12 (00000000 00000000 00000000 00001100) に変換され変数に格納される

  逆に言えば，scanf と `%d` では数字以外の入力は扱えない。
- キーボードから 1 文字入力し，それを文字のまま扱う関数として getchar がある。ただし，getchar がキーボードから受け取った文字は int 型変数に代入することになっている。

  ```
  int i;  /* getchar が取り込んだ文字は int 型変数に格納するのが決まり */
  i = getchar();  /* キーボードから入力する 1 文字を i に代入 */
  putchar(i);  /* 変数 i の内容を文字として表示 */
  ```

■リスト 5.2 **入力を出力へコピーする**

```
/* copy stdin to stdout */
#include <stdio.h>

int main()
{
     int c;

     while ((c = getchar()) != EOF)
          putchar(c);

     return 0;
}
```

実行結果例

```
abc123
abc132
def456
def456
```

- `while ((c = getchar()) != EOF) { ループ本体 }`

  while ループに入った直後 (ループ本体の実行前) に `c = getchar()` が実行され, 続いて `c != EOF` が評価される。
- `c = getchar()` — 標準入力 (キーボード) から 1 文字読み込んで，変数 c に代入する。
- `(c = getchar()) != EOF` の動作
  - getchar() の値を c に代入し，それと EOF を比較する。
  - = の優先順位は != より低いので，括弧が必要。

- 括弧がなければ,

  ```
  c = (getchar() != EOF)
  ```

  の動作をする。(c には真偽を意味する整数「非 0 or 0」が入る)

- `EOF` (End Of File)
  - 入力の終わりを表す (通常 -1 の値をもつ記号定数)。
  - `#include <stdio.h>` と書けば，プログラムで常に利用できる。
  - UNIX でキーボードから EOF を入力するには ctrl-d をタイプする (Windows では ctrl-z)。
  - 文字だけであれば char 型の変数に格納できるが，負の整数である EOF も格納するには char 型ではサイズが不十分。getchar が，キーボードから取り込んだ文字を char 型ではなく int 型として返すのは，EOF を扱うため。