補助資料

■ 文字の内部表現

コンピュータの記憶装置（メモリ）は値を 1 バイトずつ記録することしかできません。
つまり文字にせよ数値にせよ、何らかのデータに変えて記憶します。（符合化と呼ばれるプロセスです。コンピュータ概論でやった内容ですが理解していますか？）

□ 例えば文字「a」

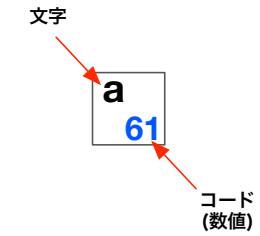
さて文字「a」は ASCII 文字コード表を見ると 16 進表記では 61 つまり 10 進での 97。つまり「a」を表現するためのデータ（値）は 97 です。

C言語での表現　　対応する文字コード
16進表記　　10進表記

$$'a' = (61)_{16} = (97)_{10}$$

ASCII 文字コード表

表の見方



ASCII文字は 1 バイトで表現されているが、その実体は数値である。
つまり 'a' は番号 61 の文字。
61 は 16 進法での表記なので、10進法では 97 番文字となる。（つまりこの 1 バイトには値 97 が入る）

・20番(16進)の sp は空白文字。
・\t はタブ。
・\n は改行文字。
・\0 はいわゆるNULL文字。
(文字列の終端記号)

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

文字型変数 c に英字 'a' を格納するコードを考えて下さい。

```
char c;
c = 'a';
```

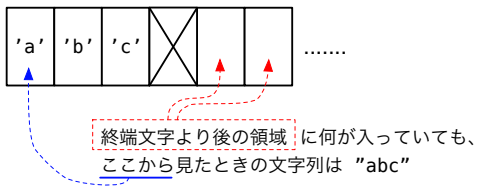
変数 c は char 型つまり長さ 1 バイトの領域を用意するものです。つまり変数 c の 1 バイトには 97 という値が入っています。実際、以下のように %d （値を 10 進表記で表示せよ）で出力すると、97 と出ます。

```
printf("%d\n", c);
```

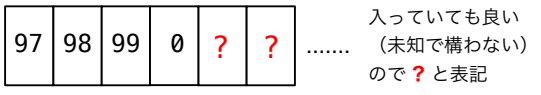
□ 文字列「abc」は

C 言語では文字列は連続するバイト列に一文字ずつ格納し、最後に終端文字を入れておくことになっています。つまり "abc" という文字列を表現するバイト列は、こんな形です。
文字 a の値は上に示したとおり 97。b は 98, c は 99 です。そして終端文字の値は 0 （ゼロ）です。上の ASCII コード表では '\0' と表記しています。C 言語では終端文字の値にゼロを使うのです。
つまり具体的なデータ並びは右のようになっています。
このように三文字を扱うための配列変数は、終端文字を含めた 4 要素で用意する必要があります。

"abc" を表現する文字列



"abc" を表現する文字列



終端より後ろは何が入っていても良い（未知で構わない）ので ? と表記

```
char s[4];
```

文字列格納用の配列変数は、処理に必要な最低量以上で宣言されれば良いだけで、例えば char s[100]; などと、より多くの要素数で用意しても構いません。終端文字さえセットしてあれば、それ以降は文字列としては扱いませんから。

□ 参考：一文字だけの文字列を扱うための配列変数

課題 1 のように、一文字だけの「文字列」を格納するためには、終端文字を含めて 2 要素用意することになります。

```
char ss[2];
ss[0] = 'a';
ss[1] = '\0';
```

"a" を格納した状態