文字列 (基本)

山本昌志*

2007 年 1 月 19 日

概 要

C 言語で文字列の処理の方法を学習する．最初に文字コードの話をして，C 言語での取扱い方法を説明する．

1 前回の復習と本日の内容

1.1 前回の復習

前回の講義では，(1) ファイルのデータを読み込む方法，(2) 配列を関数に受渡す方法を学んだ．

ファイルからのデータの読み込み ファイルからデータを取得するには，(1) ファイル情報を格納する変数を用意する，(2) ファイルをオープンする，(3) ファイルからデータを読み込む，(4) ファイルをクローズするという一連の動作が必要である．

```
FILE *in_file;

in_file = fopen("/home/yamamoto/tmp/program/int_data.txt", "r");

for(i=0; i<10000; i++){
  fscanf(in_file,"%d%d%d%d",
 &data[i][0],&data[i][1],&data[i][2],&data[i][3]);
}

fclose(in_file);
```

関数への配列データの受渡し 呼び出し側では配列名のみを，ユーザー定義関数では配列名と左端を空欄とした要素数を記述している．こうすることにより，ユーザー定義関数に配列のデータを渡すことができる．

```
hoge = cal(data);                    // 関数の呼出し

int cal(int hoge[][4]){              // 呼び出される関数
```

*独立行政法人　秋田工業高等専門学校　電気情報工学科

1.2 本日の学習内容

本日は，教科書 [1] の p.240–260 の範囲を学習する．ここで，理解すべき内容は以下のとおりである．

- コンピューターで文字を扱う場合，文字コードと呼ばれる整数として取り扱うことが分かる．
- C 言語で文字列を取り扱う場合，配列を使うことが分かる．

2 コンピューター内部での文字の表現

2.1 英数字

コンピューター内部では，全てのデータは数字として取り扱われる．文字データの場合，全ての文字について整数との対応関係が決められている．コード表と呼ばれるものが文字と整数との関係を示す．例えば，英数字の場合，アスキーコード表 (教科書 [1] の p.374) というものがあり，世界中で使われている．英数字は，この表にしたがい 0〜127 の整数が割り当てられているのである．

C 言語のプログラムで，アスキーコードの値を確かめたければ，以下のようにすればよい．文字'L' を表す整数値を表示するプログラムである．実行結果から分かるように，文字'L' は 16 進数で 4c，10 進数で 76 である．教科書 [1] の p.374 のアスキーコード表と比較せよ．

```
#include <stdio.h>
int main(void)
{
  char c;

  c='L';
  printf("%c = %x\t%d\n",c,c,c);

  return 0;
}
```

実行結果

```
L = 4c  76
```

この例で分かるように，文字型変数 (char) を用いて，文字を格納することができる．しかし，この文字型の変数は，-128〜127 まで[1]の整数しか格納できない[2]ことになっている．従って，文字型の変数で表現できるのは 256 種類の文字に限られる．アルファベットを使う文化圏では，このように 256 文字もあれば全ての文字が表現できる．0〜255 と言うのは，16 進数で 2 桁—2 進数で 8 桁—でコンピューターにとって都合が良い．

ここで，理解して欲しいのは，文字は整数に置き換えられると言うことである．コンピューター内部では，この整数を扱うことにより文字の処理をしている．そして，文字と整数の対応を示したものがコード表である．

[1]処理系によっては，0〜255 のものもある．
[2]これを 8 ビット，1 バイトと言う．詳しくは，次章の「ポインタ」で学習する．

2.2 日本語

中国語や日本語のように多くの文字を使う文化圏では，256 種類しか表現できないコードは使い物にならない．その解決のために，日本語はアスキーコードとは異なる次のような文字コードが使われる．

日本語 euc	extend unix code の略で，主に UNIX で使われる．
shift-jis	主にパソコン (DOS や Windows, macintosh) で使われている．
JIS コード	日本工業規格 JIS(Japanese Industrial Standards) が決めたコード．
ユニコード	漢字を含む世界のすべての文字を全部表現できるコード．

いろいろあるが，すべて文字と整数との対応が決められている．ただし，コード毎にその対応は異なる．例えば,「秋」という漢字は，表 1 のようになっている．英数字では 16 進数で 2 桁—1 バイト—に対して，日本語では 4 桁—2 バイト—必要である．

表 1: ”秋”をそれぞれの日本コードで表す．c 言語の場合，先頭に 0x とつく数字は 16 進数を表す．

コード	16 進数	10 進数
日本語 EUC	`0xbda9`	48553
shift-jis	`0x8f48`	36680
jis コード	???????	????????
ユニコード	???????	????????

英数字と日本語では，1 文字を表現するために必要な情報量が異なる．英数字を表す文字コードは 0〜255(1 バイト)，日本語では 0〜65535(2 バイト) が使われる．

- 英数字は，1 バイトで十分である．
- 日本語 (ユニコードを除く) は，2 バイト必要である．

3 文字の記憶方法

文字をコンピューターのメモリーに格納させるためには，その文字が持つ情報量を考えなくてはならない．先に述べたように，1 文字記憶するためには英数字では 1 バイト，日本語では 2 バイトが必要である．これは，記憶させるための箱 (変数) の大きさが，日本語では英語の 2 倍必要と言うことである．

通常の文字の処理では，1 文字では余り役に立たない．普通の文章は，文字が連なって，一つの情報の固まりとなっている．このように文字が連なったものを**文字列**と言う．この様な場合，文字型の配列を使うことになる．このようなことから，文字の状態によって，記憶方法を変える必要がある．これは，プログラマーに対して，厳しいことを要求している．ちゃんと理解するためには，少しだけハードウェアーの知識が必要になる．

ここでは，このような文字をメモリーへ格納する方法を示す．

3.1 英数字 1 文字の場合 (文字型変数)

英数字 1 文字の場合，これは単純で文字型変数を用いる．次のように宣言をすれば，変数名 hoge の文字を入れる入れ物が用意される．この入れ物の容量は，1 バイトである．

```
char hoge;
```

型名の char と言うのは，character(文字) の略である．このように宣言した文字型の変数には，ひとつの英数字が格納できる．これまで学習してきた数値と同様に，代入演算子 (=) を使う．具体的には，次のようにする．

```
hoge = 'A';
```

格納したい文字をシングルクォーテーションで囲むことを忘れてはならない．シングルクォーテーションで囲まれたひとつの英数字—ここでは A—が文字型の変数に格納される．

文字型変数のイメージを，図 1 に示す．

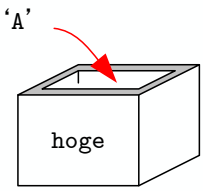

図 1: 文字型変数 hoge を用意して，それに'A' を格納．この箱の大きさは，1 バイトなので，日本語を入れることはできない．

表示まで含めた文字型変数を使ったプログラムは，次のようになる．文字型変数を表示させるためには，変換指定子%c を用いる．c は，character の略であろう．

```
#include <stdio.h>
int main(void){
  char hoge;              /* 文字型の変数 */
  hoge='A';               /* 代入 */
  printf("%c\n",hoge);    /* 表示 */
  return 0;
}
```

3.2 英数字の文字列の場合 (文字型配列)

3.2.1 英数字の文字型の配列

次に，いくつかの英数字で構成される文字列の場合である．このようなときは，文字型の配列のデータ構造を使う．以前，同じ型の数値データが複数ある場合，int 型あるいは double 型の配列を使ったのと同じである．

文字型の配列を使うときには，次のように宣言する．

```
char hoge[10];
```

これで，10 個の文字を格納できるメモリーの領域が確保できる．しかし，実際に入れることができる文字数は，9 文字である．文字列の最後に，文字列終了の記号 ('\0') を入れるため，1 文字分少なくなる．これは，アスキーコードの 0 番の NUL[3] のことである．図 2 が文字列”Akita”を文字型配列に格納するイメージである．

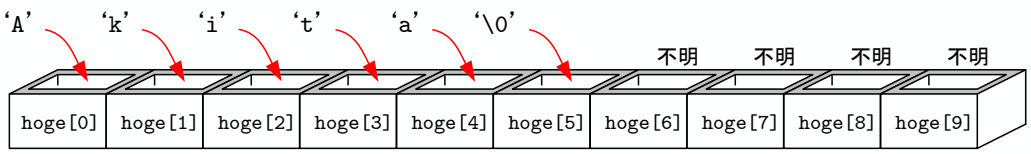

図 2: 文字型の配列 hoge[10] を用意して，それに”Akita”を格納．一つの箱には，一つの英数字しか入れられない．そして，最後に \0 が入る．

3.2.2 文字列の代入

以下に示すような方法で，この文字型の配列に文字を格納することができる．

配列の要素毎に代入 教科書 [1] の p.251 のように，配列の要素毎に文字を入れることができる．

```
hoge[0]='A';
hoge[1]='k';
hoge[2]='i';
hoge[3]='t';
hoge[4]='a';
hoge[5]='\0';
```

最後にヌル文字'\0' を入れなくてはならない．実際，この方法で文字を代入することはまれである．文字列のある特定の要素を操作するには有効である．

初期化と同時に代入 変数宣言とともに文字型の配列を初期化することができる．こうすると，配列 hoge[] には"Akita\0"が格納される．

```
char hoge[]="Akita";
```

[3] ヌル文字と呼ばれる

文字列操作関数　文字列を操作する関数を用いて，配列に文字を代入することができる．これは，教科書 [1] の p.252 に書いてある方法である．具体的には，次のようにする．

```
strcpy(hoge, "Akita");
```

これで，配列 hoge[] に文字列"Akita \0"が格納できる．ただ，この方法を使う場合は，プログラムの最初に#include <string.h>とヘッダーファイルをインクルードする必要がある．

sprintf() 関数を利用　この間数を使うとディスプレイ出力と同じように文字型の配列に代入できる．

```
sprintf(hoge, "Akita");
```

これで，配列 hoge[] に文字列"Akita \0"が格納できる．printf() や fprintf() 関数とよく似ている．これを上手に使うと，いろいろな文字列の格納が可能である．例えば，整数型の変数 nen に 2007 が格納されていたとする．次のようにすると，文字型の配列 fuga[] には，文字列”Year=2007\0”が格納される．

```
sprintf(fuga, "Year=%d",nen);
```

ポインターの利用　最後に余談であるが，ポインターというものを使えば，次のようなこともできる．簡単な文字列を代入したい場合は，sprintf() 関数よりも手軽である．

```
#include <stdio.h>
int main(void){
  char *hoge;            /* 文字型のポインターを用意 */

  hoge="Akita";          /* 代入 */
  printf("%s\n",hoge);   /* 表示 */

  return 0;
}
```

ポインターを使っても，hoge[0] とか hoge[3] のように要素毎に取り扱うことが可能である．ただし，代入はできない．ポインターについては，9 章で学習するので，ここでは，こんな方法もあるのかという程度のことで良い．

文字型配列に文字列を格納する方法を示したが，これら以外の方法も存在するであろう．sprintf() 関数とポインターを使う方法がお勧めである．

3.2.3　文字列の表示

表示まで含めた文字型配列を使ったプログラムは，次のようになる．配列に格納された文字列を表示させるためには，変換指定子%s を用いる．s は，string(ひも，一続き) の略であろう．

```
#include <stdio.h>
int main(void){
  char hoge[10];             /* 文字型の変数 */
  sprintf(hoge, "Akita");    /* 代入 */
  printf("%s\n",hoge);       /* 表示 */
return 0;
}
```

3.2.4 注意

配列を使う場合，そのサイズはプログラマーが決めなくてはならない．そのサイズを超えて，代入をするような操作をすると，エラーメッセージを出して止まる．あるいは，コンピューターがクラッシュすることもあり得る．従って，通常プログラムを作成するときには，十分大きい配列を用意するのが普通である．

配列のサイズを超えた場合，実際の動作は処理系に依存する．通常は,「segmentation fault」あるいは「セグメンテーション違反です」とかのメッセージを出して，プログラムは止まる．

3.3 日本語の場合

3.3.1 日本語の文字型の配列

文字型の場合，それに格納できるデータは 1 バイトである．一方，日本語の場合，文字は 2 バイトで表現する．文字型が 2 バイトであれば問題ないのであるが，コンピューターを発展させたのが米国であるため，仕様は 1 バイトになってしまった．そこで，日本語を扱う場合，少しばかり考えなくてはならない．

この問題を解決するために，日本語では 2 個の文字型のデータ領域に 1 個の日本語の文字を格納しているのである．なんか,「泥縄式」の解決法に思えるが，実際そうなっているのである．結構，コンピューターの世界もいい加減である．文字型の配列 hoge[] に文字列”秋田”を格納した場合，図 3 のようになる．hoge[0] と hoge[1] の 2 バイトを使って”秋”，hoge[2] と hoge[3] で”田”，hoge[0] に”\0”を格納している．

日本語は 2 バイトでひとつの文字を表すため，hoge[0]=’秋’; のような代入はできない．これには注意が必要である．

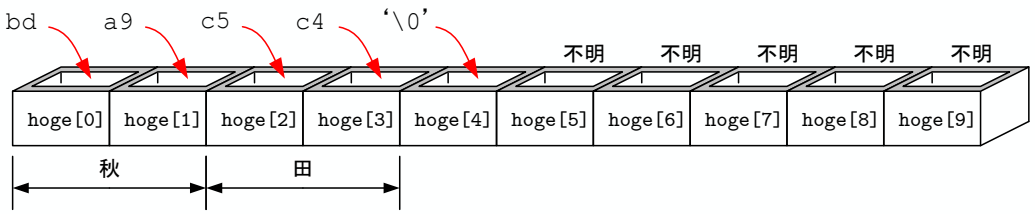

図 3: 文字型の配列 hoge[10] を用意して，それに”秋田”を格納．日本語の場合，2 つの箱で 1 つの文字が入る．そして，最後に \0 が入る．

3.3.2 文字列の代入

実際に，文字型の配列に日本語の文字を格納する方法は，アルファベットとほとんど同じである．

```
char bar[]="秋田"
strcpy(hoge,"秋田");        // char hoge[10]; のように宣言
sprintf(fuga,"秋田");       // char futa[10]; のように宣言
foo="秋田";                 // char *foo のように宣言
```

3.3.3 文字列の表示

表示方法もアルファベットと同じである．ただし，変換指定子の%cが使えないことには注意が必要である．

```
#include <stdio.h>
int main(void){
  char hoge[10];
  sprintf(hoge, "秋田");
  printf("%s\n",hoge);
  return 0;
}
```

3.3.4 注意

ここでも，配列を使うため，そのサイズに気を付ける必要がある．アルファベット同様，余裕を持って，配列を確保しなくてはならない．ただし，アルファベットと異なり日本語の場合，文字数の 2 倍と文字列の終わりを示す \0 が格納できるサイズが少なくとも必要である．

4 プログラム作成の練習

[**練習 1**] キーボードから 1 文字 (英数字) を読み込んで，その文字コードを 10 進数と 16 進数で表示するプログラムを作成せよ．

[**練習 2**] 次の動作をおこなうプログラムを作成せよ．ただし，配列へのデータの格納方法には，(1) 配列の宣言と同時に初期化，(2)`strcpy()` 関数の利用，(3)`sprintf()` 関数の利用という 3 通りの方法でプログラムを作成すること．

- 配列 `hoge` には，”Akita National College of Technology”を格納する．配列 `fuga` には，”秋田工業高等専門学校”を格納する．
- 配列 `hoge` と `fuga` に格納された文字列を表示する．

5 課題

次の講義 (1 月 26 日) の AM8:45 までに，以下の課題をレポートとして提出すること．表紙等は，いつもの通り．表紙のタイトルは「文字列 (基本)」とすること．

[**問 1**] (予復) 教科書 p.248–268 を 2 回読め．レポートには「2 回読んだ」と書け．

[**問 2**] (復) 本日配布したプリントを 2 回読め．レポートには「2 回読んだ」と書け．そして，誤字脱字，日本語の文章のおかしなところ，間違いがあれば，レポートに記述せよ．

[**問 3**] (復) 次の動作をおこなうプログラムを作成せよ．

- 配列 `hoge` には，”Department of Electrical and Computer Engineering”を格納する．配列 `fuga` には，”電気情報工学科” を格納する．

– 配列 hoge と fuga に格納された文字列を表示する.

[**問 4**] 次のプログラムの動作結果を示せ.

リスト 1: 課題のプログラム.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void)
5  {
6    char hoge[6];
7    int i;
8
9    strcpy(hoge,"Kosen");
10   for(i=0; i<6; i++){
11     printf("%c\t%d\t%x\n", hoge[i], hoge[i], hoge[i]);
12   }
13
14   return 0;
15 }
```

付録 A　参考資料

付録 A.1　アスキーコード

教科書 [1] の p.374 のアスキーコード表の見方を示す．同じもの[4]を表 2 に載せておく．この表から数字 (10 進数) を計算する方法は，次のようにする．

1. 対応する文字の上位 3 ビットの値を探す．例えば，大文字の'L' は，4 となる．
2. 次に下位 4 ビットを探す．大文字の'L' は，C である．
3. 対応する整数の計算を行う．計算方法は 16×(上位 3 ビット)+(下位 4 ビット) である．ただし，A=10, B=11,$\cdots$,F=15 である[5]．

具体的な例で表すと，大文字の'L' は，4C なので

$$16 \times 4 + 12 = 76 \tag{1}$$

となる．

表 2: アスキーコード表．表の行 (0〜7) は上位 3 ビットで，列 (0〜F) は下位 4 ビットを表す．表中の 2 文字以上のものは文字ではなく，制御コードと呼ばれる特殊文字である．

	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	‘	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	”	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	’	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

[4]09 は教科書では TAB となっているが，HT(Horizontal Tab) と書くことが多い．また，5C は教科書では円記号となっているが，これはバックスラッシュ (\) と同じ取扱いである．そもそも，アスキーコードを作ったアメリカでは円記号は不要である．

[5]0〜F まで用いて数値を表す方法を 16 進数と言う．ポインターを学習するときに詳細を説明する．

付録 A.2　バイトとビット

ここでは，位取り記数法の話を少ししておく．詳細は「8 章 ポインター」で学習するので，雰囲気が分かれば良い．

人間は 10 進数を使うが，コンピューター内部では 2 進数が使われている．また，実際のプログラマーは，2 進数との対応がつきやすい 16 進数を好んで使う．それぞれの対応は，表 3 のようになる．

ここでは，次のことを理解しなくてはならない．

- 2 進数の 1 桁を 1 ビット (bit) と言う．これで，0 と 1 の整数を表すことができる．
- 2 進数の 8 桁，即ち 8 ビットで，1 バイト (byte) になる．これは，10 進数だと 0〜255 までの整数，16 進数だと 0〜`ff` までの整数を表すことができる．
- 2 バイトの場合，10 進数だと 0〜65535 までの整数，16 進数だと 0〜`ffff` までの整数を表すことができる．

表 3: 10 進数と 2 進数，16 進数の関係

10 進数	2 進数	16 進数	10 進数	2 進数	16 進数
0	0	0	16	10000	10
1	1	1	17	10001	11
2	10	2	18	10010	12
3	11	3	19	10011	13
4	100	4	20	10100	14
5	101	5	21	10101	15
6	110	6	22	10110	16
7	111	7	23	10111	17
8	1000	8	24	11000	18
9	1001	9	25	11001	19
10	1010	a	26	11010	1a
11	1011	b	⋮	⋮	⋮
12	1100	c	255	11111111	`ff`
13	1101	d	⋮	⋮	⋮
14	1110	e	65535	1111111111111111	`ffff`
15	1111	f	⋮	⋮	⋮

参考文献

[1] 内田智史監修, (株) システム計画研究所編. C 言語によるプログラミング　基礎編　第 2 版. (株) オーム社, 2006.