10.文字列

■ 文字列の取り扱い

文字列を扱うには、図1に示すように**文字型の配列**として宣言する。

```
#include<stdio.h>
int main(void){
        char str_a[] = {'a', 'b', 'c', '¥0'}; //'¥0'に注意
        char str_b[] = "def";
        int i;
        printf("str_a = %s¥n", str_a);
        printf("str_b = %s¥n", str_b);

        for(i=0; i<3; i++){
                printf("c) str_a[%d] = %c¥n", i, str_a[i]);
                printf("x) str_a[%d] = 0x%x¥n", i, str_a[i]);
        }
}
```

文字列は文字型の配列

```
str_a = abc
str_b = def
c) str_a[0] = a
x) str_a[0] = 0x61
c) str_a[1] = b
x) str_a[1] = 0x62
c) str_a[2] = c
x) str_a[2] = 0x63
```

実行結果

図1　文字列を利用したプログラム例

■ 文字列の仕組み

図1に示すプログラムの文字配列 str_a[ ]と str_b[ ]と、メモリとの関係を図2に示す。「1文字が1配列変数に入る」こと、「文字列の終わりには終了記号' ¥0' が入る」ことに注意する。

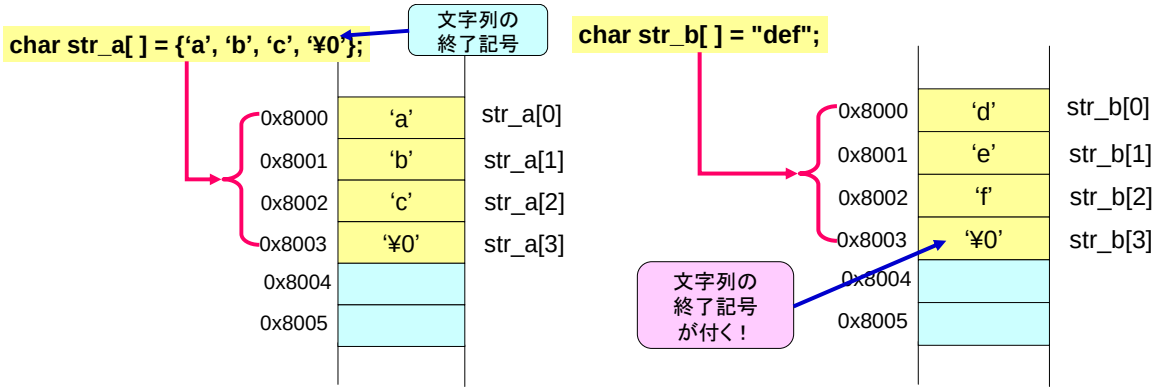

図2　文字列とメモリ(1)

◎ ASCII コード

メモリには、図3に示すとおり、実際には文字を表す数値（文字コード）が入る。' a' ならば、0x61 がメモリに入る。文字と数値の対応には、様々な規格があるが、C 言語の多くの処理系では、ASCII コードが用いられている。

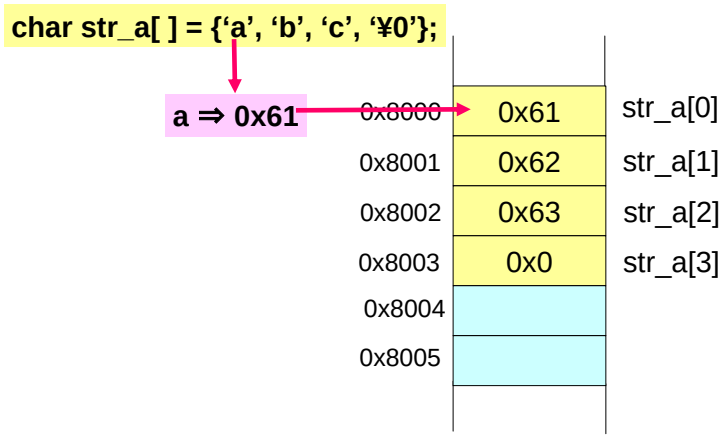

図3　文字列とメモリ(2)

■ 文字列の利用方法

◎ 宣言と初期化

(1) 宣言方法と初期化には、下に示すいくつかの方法がある。配列の大きさが「文字列の長さ＋1」になることに注意。

```
char str[6] = { 'h', 'e', 'l', 'l', 'o', '¥0' };      ※1 文字ずつ、1 つの配列変数に入れる(最後は必ず¥0)
char str[] = { 'h', 'e', 'l', 'l', 'o', '¥0' };
char str[6] = "hello";                                 ※文字列を一度に入れる(¥0 は入れない)
char str[] = "hello";
```

¥0

(2) 文字列を 2 次元配列で宣言して、初期化する場合

```
char nengou[][7] = {"heisei", "shouwa"};      ※[ ][7]の 7 は文字列の長さ
```

◎ 入出力

(1) 文字列の入力

```
char str[ ] ="hello";
str[1] = 'p' ;
scanf("%s", str);
```

scanfでも**%s**を使用
&がないことに注意

(2) 文字列の出力

```
char str[6];
printf("%s", str);
```

printfには**%s**を使用

■ 文字列で多い誤り

文字列は 1 文字ずつ配列要素として格納されていることに注意。従って、下のようなことは**できない**。

- 「文字列を比較したい」場合

```
char str_a[ ] = "abc";
char str_b[ ] = "def";
if (str_a = = str_b)
```

str_aとstr_bは、配列の先頭アドレスが違うので、比較できない

- 「変数に文字列を保存したい」場合

```
char str_a[4] ;
str_a = "abc"
```

文字列配列に代入できるのは初期化の時のみ

- 文字列配列の大きさに「文字数」は指定できない

```
char str_a[3]="abc"
```

配列の大きさは、「文字数＋1」

- scanf に&を付けない

```
scanf("%s", &str);
```

■ 文字列を操作するためのライブラリ関数

前ページの誤りで記したとおり、文字列は直接比較したり、入力したりすることができない。そこで、下記の関数が標準で用意されている。

※下記の関数を利用するためには、#include<string.h>をプログラムの先頭に追加する。

文字列の比較	
strcmp(str1, str2)	str1 と str2 が文字列配列の時、str1 と str2 を比較し、同じならば、0 を返す。

	例）
	char str1[ ] = “abc”;
	char str2[ ] = “abc”;
	if (strcmp(str1, str2) = = 0){
	printf(“str1 と str2 は同じ”)
	}

strncmp(str1, str2, n)	str1 と str2 が文字列配列の時、str1 と str2 を先頭から n 文字分比較し、同じならば、0 を返す。

	例）
	char str1[ ] = “abc”;
	char str2[ ] = “abcder”;
	if (strncmp(str1, str2, 3) = = 0){
	printf(“str1 と str2 は同じ”)
	}

文字列のコピー	
strcpy(dst, src)	src と dst が文字列の時、src を dst にコピーする。
	例 1） char src[ ] = “abc”; char dst[4] ; strcpy(dst, src);　//文字列 dst に”abc”がコピーされる
	例 2） char str[4]; strcpy(str, “abc”); //文字列 str に”abc”が代入される
strncpy(dst, src, n)	src と dst が文字列の時、src を dst に n 文字分コピーする。

	例 1）
	char src[ ] = “abc”;
	char dst[ ] = “defghi”;
	strcpy(dst, src,3);
	//文字列 dst に 3 文字分コピーされ、”abcghi”となる

	例 2） char str[4]; strcpy(str, “abcdefg”, 3); //文字列 str に 3 文字分コピー”abc”が代入される