文字・文字列処理関数

山本昌志*

2007 年 1 月 26 日

概 要

C 言語での文字や文字列の処理方法を学ぶ．はじめに，文字列処理のユーザー定義関数を作成し，文字列の取扱いに慣れる．そして，文字・文字列処理のライブラリー関数の使い方の練習を行う．

1 前回の復習と本日の内容

1.1 前回の復習

- コンピューターで文字を扱う場合，文字は文字コードで決められた整数として扱われる．一般に英数字の場合，アスキーコードが使われる．日本語の場合，EUC や S-JIS コードが使われることが多い．これらのコードは，文字と整数との対応を表す．
- 英数字は 1 バイトで表現可能である．それに対して，日本語は 2 バイト必要である．
- C 言語で文字列を取り扱う場合，文字型の配列を使う．

1.2 本日の学習内容

教科書 [1] の p.256–268 が，本日の学習範囲である．学習のゴールは，次の通り．

- 文字処理を行うユーザー定義関数の作成ができること．
- 文字や文字列の処理のためのライブラリー関数が使えること．

2 文字列コピー関数

教科書 [1] の p.256–268 には，文字列を処理する次のようなユーザー定義関数を示して，その作成方法を記述している．

- 文字列をコピーする関数 `str_cpy()`
- 文字列の長さを計算する関数 `str_length`
- 文字列を連結する関数 `str_cat()`

*独立行政法人　秋田工業高等専門学校　電気情報工学科

- 文字列を比較する関数 str_comp

これらのうち，str_cpy() を例にして，文字列の取り扱い方を説明する．他は，説明の時間がないので，各自，教科書を読んで理解せよ．

教科書と同じ関数だと面白くない[1]．そこで，リスト 1 のように，関数 my_str_cpy() [2]という関数を作成した．これは，文字列をコピーして，コピーしたバイト数— \0 を含まない—を返す関数である．プログラムの内容は，これまでに学習した範囲で理解できるはずである．大事な点は，以下の通り．

- ユーザー定義関数へ文字型の配列の情報を実引数として渡すときには，呼出元では配列名のみ書く．これは，数値の場合と同じ．
- 配列を受け取るユーザー定義関数の仮引数は，配列の型と配列名，サイズを書く．ただし，配列のサイズの左端は書かない．これも，数値の場合と同じ．
- 配列の場合，呼出元と呼び出された関数では同じメモリーを使う．呼び出された関数で配列を書き換えたら，呼出元の配列も書き変わる．これも，数値の場合と同じ．
- 文字列の終わりを示す記号 \0 を目印にして，配列の要素ひとつずつコピーしている．

リスト 1: 文字列をコピーして，バイト数を返す関数の例

```
1  #include <stdio.h>
2
3  int my_str_cpy(char dest[], char src[]);
4  //================================================
5  // メイン関数
6  //================================================
7  int main(void){
8    char foo[30]="おもろいことないかー";
9    char hoge[30], fuga[30];
10   int a, b;
11
12   a=my_str_cpy(hoge, foo);
13   b=my_str_cpy(fuga, "情報処理が，ぼちぼちやでー");
14   printf("%dバイトコピー\t%s\n", a, hoge);
15   printf("%dバイトコピー\t%s\n", b, fuga);
16
17   return 0;
18 }
19
20 //================================================
21 // 文字列のコピー関数
22 //================================================
23 int my_str_cpy(char dest[], char src[])
24 {
25   int i=0;
26
27   while(src[i]!='\0'){
28     dest[i] = src[i];
29     i++;
30   }
31
32   dest[i]='\0';
33
34   return i;
```

[1]同じプログラムだと，著作権の問題もからむ．

[2]my は「私の」と，私のイニシャル (Masashi Yamamoto) から名付けている．

```
35 }
```

実行結果

```
20 バイトコピー　おもろいことないかー
26 バイトコピー　情報処理が，ぼちぼちやでー
```

3　文字・文字列処理関数

文字や文字列の処理のプログラムを容易にするために，標準ライブラリー関数[3]が用意されている．これを使えば，いちいちユーザ定義関数を作成するまでもなく，文字・文字列処理ができる．教科書のユーザー定義関数と同じ働きの標準ライブラリー関数もある[4]．文字．文字列処理のライブラリー関数を付録付録 A に載せる．

諸君は，これらのライブラリー関数を全て憶える必要はない．ライブラリー関数の大体の機能とその使い方が書かれている場所さえ知っていれば良い．プログラムを作成するときには，必要なライブラリー関数は C 言語の本，あるいは WEB から探すことになる．私のテストでも，これらのライブラリー関数は与えるので憶える必要はない．

それでは，文字処理と文字列処理のライブラリー関数の使い方を学ぶことにしよう．

3.1　文字処理関数の例

付録付録 A の表 1 に示す文字処理のライブラリー関数の使い方を示す．この表の関数は，ひとつの文字を処理する関数である．文字列ではない．ひとつの文字というのは 1 バイトのことで，2 バイトで表す日本語—ひらがなやカタカナ，漢字—の場合，この文字処理関数を使うことはできない．

この文字処理関数のうち，文字が英数字[5]か否かを調べる関数 `isalnum()` の使用例をリスト 2 示す．キーボードからひとつの文字を読み込んで，それが英数字か否かを調べ，その結果を表示している．

表 1 に示すとおり，便利な文字列処理関数はたくさんある．それを使うためには，`ctype.h` というヘッダーファイル[6]をインクルードしなくてはならない．ただし，コンパイル (`gcc`) 時には，特別なオプションは必要ない．

リスト 2: 入力された文字が英数字か否かを判断する．

```
1 #include <stdio.h>
2 #include <ctype.h>
3
4 int main(void)
5 {
6   char hoge;
7
8   scanf("%c", &hoge);
```

[3]ANSI 規格の C 言語で予め用意されている関数群.
[4]教科書の著者はもちろんこのことは知っている．教科書は学習のために，わざわざユーザー定義関数を作成している
[5]アルファベットの a–z と A–Z，数字の 0–9 のこと.
[6]関連した関数に関する宣言やマクロがかかれたファイル.

```
 9
10    if(isalnum(hoge)){
11      printf("入力された文字は英数字です．\n");
12    }else{
13      printf("入力された文字は英数字ではありません．\n");
14    }
15
16    return 0;
17  }
```

3.2 文字列処理関数の例

付録付録 A の表 2 に示す文字列処理のライブラリー関数の使い方を示す．この表の関数は，文字列を処理する関数である．

この文字列処理関数のうち，文字列をコピーする関数 `strcpy()` の使用例をリスト 3 に示す．このプログラムを実行してみると，文字列を配列にコピーしていることが分かるだろう．

この例からも分かるように，戻り値は使わなくてもよい．文字列処理関数の場合戻り値がポインターの場合が多いので，まだ，諸君には難しいであろう．次回の講義からポインターの話をするので楽しみにしてください．

表 2 に示すとおり，たくさんの便利な文字列処理関数がある．それを使うためには，`string.h` というヘッダーファイルをインクルードしなくてはならない．ただし，コンパイル (gcc) 時には，特別なオプションは不要である．

リスト 3: 文字列をコピーする関数 `strcpy()` の使用例．

```
1   #include <stdio.h>
2   #include <string.h>
3
4   int main(void){
5     char foo[30]="もうかりまっか?";
6     char hoge[30], fuga[30];
7
8     strcpy(hoge, foo);
9     strcpy(fuga, "ごっつい，もうかるでー");
10    printf("%s\n", hoge);
11    printf("%s\n", fuga);
12
13    return 0;
14  }
```

3.3 文字列処理関数の例

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，それが英文字か否かを表示するプログラムを作成せよ．

[**練習 2**] キーボードから 1 文字 (英小文字) を読み込んで，その大文字を表示するプログラムを作成せよ．

[**練習 3**] 二つの文字列を連結する関数を作成し，実行せよ．ただし，関数の戻り値の型は，`void` でよい．

[**練習 4**] ライブラリー関数 `strcat()` を使うプログラムを作成せよ．

[**練習 5**] ライブラリー関数 `strlen()` を使うプログラムを作成せよ．

5 課題

次の講義 (2 月 2 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・文字列処理関数」とすること．

[**問 1**] (予復) 教科書 p.258–292 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) 文字列を連結して，連結後の文字数を返すユーザー定義関数を作成せよ．

[**問 4**] (復) ライブラリー関数 `strncpy()` を使うプログラムを作成せよ．

付録 A　文字・文字列処理のライブラリー関数

付録 A.1　文字処理関数

表 1: 文字処理関関数．#include <ctype.h>が必要．変数は，int c;.

関数名	動作	戻り値
isalnum(c)	英数字なら真	真/偽 (整数型)
isalpha(c)	英文字なら真	真/偽 (整数型)
iscntrl(c)	制御文字なら真	真/偽 (整数型)
isdigit(c)	数字なら真	真/偽 (整数型)
isgraph(c)	印字可能文字なら真	真/偽 (整数型)
islower(c)	小文字なら真	真/偽 (整数型)
isprint(c)	空白以外の印字可能文字なら真	真/偽 (整数型)
ispunct(c)	区切り文字なら真	真/偽 (整数型)
isspace(c)	空白類文字なら真	真/偽 (整数型)
isupper(c)	大文字なら真	真/偽 (整数型)
isxdigit(c)	16 進表示文字なら真	真/偽 (整数型)
tolower(c)	文字 c を小文字に変換	小文字 (文字型)
toupper(c)	文字 c を大文字に変換	大文字 (文字型)

付録 A.2　文字列処理関数

表 2 を使うためには，#include <string.h>が必要である．変数は，char s1[256],s2[256]; のように文字型の配列．そのサイズは，処理に必要なサイズよりも大きいこと (256 とは限らない)．後の学習範囲であるが，s1 や s2 は文字型のポインターでも良い．ま た，ダブルクォーテーションで囲んだリテラル表現も可能な部分もある．c は文字型の変数，char c; である．

表 2: 文字列処理関関数.

関数名	動作	戻り値
strlen(s1)	文字列 s1 の長さ，すなわち文字数を整数値返す.	文字列長 (整数型)
strcpy(s1,s2)	s1 に，文字列 s2 をコピーする.	ポインター s1 の値
strcat(s1,s2)	文字列 s1 の後に，文字列 s2 をコピーする.	ポインター s1 の値
strcmp(s1,s2)	文字列 s1 と s2 を比較する. s1 > s2 の場合，戻り値は正 s1 == s2 の場合，戻り値は 0 s1 < s2 の場合，戻り値は負	整数値
strncpy(s1,s2,n)	s1 に文字列 s2 の先頭から n 文字をコピーする.	ポインター s1 の値
strncat(s1,s2,n)	文字列 s1 の後にと文字列 s2 の先頭から n 文字を連結する.	ポインター s1 の値
strncmp(s1,s2,n)	文字列 s1 と文字列 s2 の先頭から n 文字を比較する．比較の結果は，srcmp と同じ.	整数値
strchr(s1,c)	文字列 s1 の中の文字 c の位置を返す．文字がないときは，NULL を返す.	ポインター
strstr(s1,s2)	文字列 s1 の中にある文字列 s1 の位置を返す．もし，文字列がない場合，NUL を返す.	ポインター

ここで，戻り値がちょっと難しい strchr() 関数の使用例をリスト??示す．このプログラムの動作はポインターを学習しないと理解できないであろう．したがって，今は分からなくてもよい．ただし，学年末には理解してほしい．

リスト 4: 関数 strchr() をつかって，文字のある位置を捜している．

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void)
5  {
6    char s='s';
7    char alphabet[50]="abcdefghijklmnopqrstuvwxyz";
8
9    printf("%cは%d番目です\n",s, strchr(alphabet,s)-alphabet+1);
10
11   return 0;
12 }
```

実行結果

```
s は 19 番目です
```

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.