文字処理 (その 2)

山本昌志*

2004 年 12 月 17 日

1　復習と本日の学習内容

1.1　復習

前回の授業では、

- コンピューター内部での文字の取り扱い。
- 文字型の変数。
- 文字型変数への代入方法.

を学習した。

1.2　本日の内容

本日は、前回に引き続き、文字の処理を学習する。その内容は、

- 標準入出力 (キーボード、ディスプレイ) への文字の処理。
- 文字のファイル処理。
- 文字を取り扱うライブラリー関数。

である。

2　標準入出力

UNIX の世界では、一般的に、標準入力装置としてキーボード[1]が、標準出力装置としてディスプレイが割り当てられている。文字をキーボードから、入力して、ディスプレイに書き出す方法を示す。

*国立秋田工業高等専門学校　電気情報工学科
[1]コンソールと言うこともある。

2.1 標準 (キーボード) 入力

2.1.1 getchar

まずは、キーボードから 1 文字を入力する「getchar」を紹介する。

getchar

機　　能	キーボードから 1 文字入力する。
ヘッダ ー	stdio.h
形　　式	int getchar(void)
返 却 値	読み込んだ文字を返す。ファイル終了時は EOF を返す

形式のところに書かれている void とは、実引数がないということを表している。void を日本語に訳すと「空の」という意味になる。また、実引数というのは関数の変数みたいなものである。この関数を使って、キーボードから入力された 1 文字を変数 hoge に格納するためには、次のように記述する。

```
int hoge;              /* 整数型の変数 */
hoge=getchar();        /* 代入 */
```

ここで、変数の型として int を用いているが、char としても何も問題はない。c 言語では、1 文字は整数とほとんど同じように扱われる。また、以前も述べたように、日本語のように 2 バイト文字は代入できないことに注意が必要である。この関数と対になっているのが、後で述べる「putchar」である。

2.1.2 gets

次にキーボードから 1 行読み込む関数である。1 行というのは、「Enter」キーが押されるまでである。

gets

機　　能	キーボードから 1 行読み込み、hoge(配列) に格納。
ヘッダ ー	stdio.h
形　　式	char gets(char *hoge)
返 却 値	入力文字をそのまま返す。ファイル終了時、あるいはエラーのときは NULL を返す。

この関数は、つぎのようにつかう。

```
char hoge[10];             /* 整数型の変数 */
gets(hoge);                /* 代入         */
```

この関数は便利で、最後の文字列の後に、ちゃんとその区切りの「\0」が配列に入れられる。しかし、配列で確保された文字数以上を入力すると、とんでもないこと[2] が生じる可能性があり、使う場合は気をつけなくてはならない。。そのため、コンパイラーによっては、「この関数は、危険だから使うな」とメッセー

[2] 予約した配列のメモリー以外の場所にデータを書き込んでしまう。

ジを出すものもある。後でのべる「fgets」で入力文字数を制限する方が良い。この関数と対になっているのが、後で述べる「puts」である。

2.1.3 scanf

最後に説明するのが、おなじみの「scanf」である。

scanf

機　　能	キーボードから文字列を読み込み、hoge(配列) に格納。
ヘッダ ー	stdio.h
形　　式	int scanf("%s", hoge)
返 却 値	通常、入力項目を返す。異常時は EOF を返す。

この関数は次のように使う。

```
char hoge[10];              /* 整数型の変数 */
scanf("%s",hoge);              /* 代入 */
```

この関数は、空白を入力できないという問題を抱えている。なぜならば、空白は文字列の区切りを示すので、そこで文字列が終わると判断されるからである。この関数と対になっているのが、おなじみの「printf」である。

2.1.4 fgets を使う方法

どの関数にも問題があるが、もっとも良い方法は、「fgets」を使う方法である。これは次のように書けば良い。

```
char hoge[10];              /* 整数型の変数 */
fgets(hoge,10,stdin);              /* 代入 */
```

これで、空白も入力できるし、10 文字を越えても、メモリーの内容を破壊することは無い。stdin と言うのは、standard input の略で、標準入力 (キーボード) を示す。

2.2 標準 (ディスプレイ) 出力

標準入力に対応した標準出力の命令がある。対応していると言っても、実際には関係があるわけではないが、よく似ている命令なので覚えやすい。

2.2.1 putchar

まずは、ディスプレイに 1 文字を出力する「putchar」を説明する。

putchar

機　　能	ディスプレイに 1 文字出力する。
ヘッダ ー	stdio.h
形　　式	int putchar(int hoge)
返 却 値	出力した文字を返す。異常時は EOF を返す

この関数を使って、変数 hoge に格納された文字を出力するためには、次のように記述する。

```
int hoge;              /* 整数型の変数 */
hoge=getchar();        /* 代入         */
putchar(hoge);         /* 1 文字出力   */
```

「putchar」でディスプレイに文字を書いた場合、改行の「\n」 は出力されない。改行したい場合は、もう 1 行余分に printf などを書かなくてはならない。

2.2.2 puts

もう予想がつくと思うが、次は「puts」である。

puts

機　　能	ディスプレイに文字列を出力後、改行する。
ヘッダ ー	stdio.h
形　　式	int puts(char *hoge)
返 却 値	正常時は非負、異常時は EOF を返す。

この関数は、つぎのようにつかう。

```
char hoge[10];             /* 整数型の変数 */
gets(hoge);                /* 代入         */
puts(hoge);                /* 1 行出力     */
```

この関数は便利で、最後の文字列の後に、ちゃんとその改行「\n」が出力される。また、変換処理がないので、printf 関数よりも処理が早い。

2.2.3 printf

最後は、もう説明する必要がないと思うが、printf 関数である。

printf

機　　能	変換書式に従いディスプレイに書き出す。
ヘッダ ー	stdio.h

形　　式	`int printf("%s", hoge)`
返 却 値	正常時は出力されたバイト数を返す。異常時は負の値を返す。

この関数は次のように使う。

```
char hoge[10];              /* 整数型の変数     */
scanf("%s",hoge);           /* 代入             */
printf("%s",hoge);          /* ディスプレイ出力 */
```

もちろん、変換指定子のところに%c を使えば、文字列ではなく、1 文字を扱うことができる。

3　ファイル処理

文字の処理は、なんと言っても、ファイルと密接に関係する。文字データの多くは、キーボードから、入力するのではなく、ファイルに書かれているの普通である。人間が蓄積してきた多くの情報は文字で書かれ、現在、大部分のものが電子化、即ち、電子ファイルに保存されている。そのデータを処理するために、いちいちキーボードから入力したのでは、日が暮れてしまう。そこで、文字データをファイルから読み込むことが必要になる。ここでは、キーボードからの入力とあわせて、ファイルから文字を入力する方法を学習する。

現代社会において、情報は電子化されなくてはならない。電子化されると多くの媒体で公表され、世界中の人が使える。私の講義ノートも多くの人に見てもらうために、電子化している。インターネットを使って、世界中、どこでも私の講義ノートを見ることができる。本当に、その技術はすばらしいと思う。ただ、私の講義ノートの善し悪しは別の問題である。

3.1　ファイル入力

3.1.1　fgetc

まずは、ファイルから 1 文字を入力する「fgetc」を紹介する。

fgetc

機　　能	ファイルから 1 文字入力する。
ヘッダ ー	stdio.h
形　　式	`int fgetc(FILE *fp)`
返 却 値	読み込んだ文字を返す。異常時は EOF を返す

読み込み先のファイルは、fgetc 関数の実引数に書く。もちろん、これは open 関数により開かれたファイルのファイル記述子である。この関数の動作は、キーボードから 1 文字入力する getchar に非常によく似ている。事実、ファイル記述子を「stdin」とすれば、getchar と全く同じ働きをする。stdin とは standard input の略で、標準入力すなわち、キーボードのことである。UNIX では、キーボードもファイルと同じように取り扱われる。この関数は、次のように使う。

```
FILE *fp;
int hoge;                       /* 整数型の変数       */
fp=fopen("text.txt","r");       /* ファイルのオープン */
hoge=fgetc(fp);                 /* 1 文字代入         */
```

この関数は、後で述べる「fputc」と対の関係になっている。

3.1.2 fgets

次にファイルから 1 行を読み込む関数、fgets を示す。これは、1 行を読み込むが、指定したバイト数を越えるとそこで、読み込みを停止する。

fgets

機　　能	ファイルから 1 行読み込み、配列に格納。
ヘッダ ー	stdio.h
形　　式	char fgets(char *hoge, int n, FILE *fi)
返 却 値	入力文字をそのまま返す。ファイル終了時、あるいはエラーのときは NULL を返す。

あらかじめオープンされたファイルから、1 行を読み込み、配列 hoge に格納する。ただし、n バイト越えると読み込みは止める。この関数は、次のようにして使う。

```
FILE *fi;
char hoge[256];                 /* 文字型の配列       */
fi=fopen("rtest.txt","r");      /* ファイルのオープン */
fgets(hoge,256,fi);             /* 1 行代入           */
fclose(fi);
```

読み込むバイト数を指定しているため、gets のように、許可されていないメモリー領域の内容を書き換えるようなことはない。

3.1.3 fscanf

最後のファイル入力は、fscanf である。

fscanf

機　　能	ファイル (fi) から文字列を読み込み、hoge(配列) に格納。
ヘッダ ー	stdio.h
形　　式	int fscanf(fi,"%s", hoge)
返 却 値	通常、入力項目を返す。異常時は EOF を返す。

この関数は次のように使う。

```
FILE *fi;
char hoge[256];                    /* 文字型の配列       */
fi=fopen("rtest.txt","r");         /* ファイルのオープン */
fscanf(fi,"%s",hoge);             /* 文字列代入         */
fclose(fi);
```

この関数も、scanf 同様に空白を入力できないという問題がある。理由も同じである。この関数と対になっているのが、おなじみの「fprintf」 である。

3.2 ファイル出力

ファイル入力に対応したファイル出力の関数である。この関数は、標準出力と非常によく似ている。

3.2.1 fputc

まずは、ファイルに 1 文字を出力する「fputc」を説明する。

fputc

機　　能	ファイルに 1 文字出力する。
ヘッダ ー	stdio.h
形　　式	int fputc(int hoge, FILE *fo)
返 却 値	出力した文字を返す。異常時は EOF を返す

この関数を使って、変数 hoge に格納された文字をファイルに出力するためには、次のように記述する。

```
FILE *fi,*fo;
int hoge;                       /* 整数型の変数                 */
fi=fopen("rtest.txt","r");      /* 読み込み用ファイルのオープン */
fo=fopen("wtest.txt","w");      /* 書き込み用ファイルのオープン */
hoge=fgetc(fi);                 /* 1 文字読み込み               */
fputc(hoge,fo);                 /* 1 文字書き出し               */
fclose(fo);                      /* 読み込み用ファイルのオープン */
fclose(fi);                      /* 書き込み用ファイルのオープン */
```

「fputc」でファイルに文字を書いた場合、改行の「\n」 は出力されない。実際、改行されたら、非常に困る。

3.2.2 fputs

文字列を出力するためには、「fputs」をつかう。

fputs

機　　能	ファイルに文字列を出力する。改行は付加しない。
ヘッダ ー	stdio.h
形　　式	int fputs(char *hoge, FILE *fo)
返 却 値	正常時は非負、異常時は EOF を返す。

この関数は、つぎのようにつかう。

```
FILE *fi,*fo;
char hoge[256];                 /* 文字型の配列                  */
fi=fopen("rtest.txt","r");      /* 読み込み用ファイルのオープン  */
fo=fopen("wtest.txt","w");      /* 書き込み用ファイルのオープン  */
fgets(hoge,256,fi);             /* 1 行の読み込み                */
fputs(hoge,fo);                 /* 1 行の書き出し                */
fclose(fo);                     /* 書き込み用ファイルのクローズ  */
fclose(fi);                     /* 読み込み用ファイルのクローズ  */
```

この関数は便利で、最後の文字列の後に、ちゃんとその改行「\n」が出力される。1 行の処理をする関数なので、当たり前である。また、変換処理がないので、printf 関数よりも処理が早い。

3.2.3 fprintf

最後は、もう説明する必要がないと思うが、fprintf 関数である。

fprintf

機　　能	変換書式に従いファイル (fo)) に書き出す。
ヘッダ ー	stdio.h
形　　式	int fprintf(fo,"%s", hoge)
返 却 値	正常時は出力されたバイト数を返す。異常時は負の値を返す。

この関数は次のように使う。

```
FILE *fi,*fo;
char hoge[256];                 /* 文字型の配列                  */
fi=fopen("rtest.txt","r");      /* 読み込み用ファイルのオープン  */
fo=fopen("wtest.txt","w");      /* 書き込み用ファイルのオープン  */
fscanf(fi,"%s",hoge);           /* 1 文字列の読み込み                */
fprintf(fo,"%s",hoge);          /* 1 文字列の書き出し                */
fclose(fo);                     /* 書き込み用ファイルのクローズ  */
fclose(fi);                     /* 読み込み用ファイルのクローズ  */
```

4　文字処理のための標準ライブラリー関数

4.1　文字処理

文字の処理のために、表 1 のようなライブラリー関数が、c 言語には用意されている。プログラムを作成するときに、つかう。ただし、これを覚える必要はない。こんなものがあった程度でよい。

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

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

4.2　文字列処理

文字列の処理のために、表 2 のようなライブラリー関数が、c 言語には用意されている。プログラムを作成するときに、つかう。ただし、これを覚える必要はない。こんなものがあった程度でよい。

表 2: 文字列処理関関数。#include <string.h>が必要。変数は、char s1[256],s2[256]; のように文字型の配列。配列のサイズは、処理に必要なサイズよりも大きいこと (256 とは限らない)。c は文字型の変数、char c; である。

関数名	動作
strlen(s1)	文字列 s1 の長さ、すなわち文字数を整数値返す。
strcpy(s1,s2)	s1 に、文字列 s2 をコピーする。
strcat(s1,s2)	文字列 s1 の後に、文字列 s2 をコピーする。
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
strncpy(s1,s2,n)	s1 に文字列 s2 の先頭から n 文字をコピーする。
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。
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。
strchr(s1,c)	文字列 s1 の中の文字 c の位置を整数で返す。文字がないときは、NULL を返す。
strstr(s1,s2)	文字列 S1 の中にある文字列 S1 の位置を整数で返す。もし、文字列がない場合、NULL を返す。

5 予習

来週は、情報処理センターで実習を行う。教科書の p.236 の練習問題を中心にプログラムの作成を行う。予習をしてくること。