基礎プログラミング演習 II 教材　(#21)

■ 文字型

★教科書 p.147, 9.1「文字型のデータ」を参照
押さえて欲しいポイント：
・文字とは「a」「1」など一文字のことを指す（数字の 1 を表現する文字もある）
・文字を扱うために char 型が用意されている
・文字型定数は `'a'` などシングルクォートで囲む
・エスケープ文字による特殊記号の表現ができる

char 型の変数を宣言し、そこに文字「a」を代入するには以下のように書きます。

```
char letter;
letter = 'a';
```

□ printf, scanf での変換文字

printf 関数では%d, %f などの変換文字について説明しました。文字型の変換文字は %c です。scanf 関数でも同様に %c となります。

```
scanf("%c", &letter);
printf("%c\n", letter);
```

のように書きます。（実は printf で用いる制御文字（例えば「\n」）はエスケープ文字でした！）

■ 文字列の扱い

★教科書 p.165, 10.1「文字列の取り扱い」を参照
押さえて欲しいポイント：（p.165～167 まで）
・文字列とは「abc」「word sample」など複数の文字の集まりを指す
・文字列型はなく、文字型 char の配列として処理する
・終端文字（C 言語ではヌル文字）に注意
・変数として用意する文字型の配列は、必要な文字数+1 の長さを用意する（ヌル文字のため）

□ 文字列定数

★教科書 p.168, 10.1.3「文字列定数と配列の初期化」を参照
押さえて欲しいポイント：（p.168～169 まで）
・文字列型の変数はないが、文字列定数はある
・文字列定数はヌル文字のために一バイト長く用意される
・文字列を扱う機能は関数として実現されている（代入はできない。連結演算子はない。）
・文字配列の初期化の方法に注目（p.169 文法 10.3 参照）

□ printf, scanf での変換文字

文字列に対する変換文字は %s です。（教科書 p.170 図 10.4 参照）

```
char string[100];
scanf("%s", string);
printf("%s\n", string);
```

string は要素数 100 としてみました。これが溢れると実行時エラーとなる場合があるので宣言する要素数には注意を払うよう習慣づけましょう。配列に対する scanf では &string とはせず、単に string と書きます。理由は配列とポインタの関係を学ぶまで分からないでしょうから、いまは丸覚えして下さい。

□ 実験：文字数をカウントする

右のプログラムは scanf によって入力された文字列の文字数を出力するものです。内容を吟味して実行してください。

```
char string[100];
int length;

printf("string >> ");
scanf("%s", string);

length=0;
while( string[length] != '\0' ) {
  length++;
}
printf("%d\n", length);
```

★教科書 p.172 図 a, b を参照

（自分の想像どおりの結果が出ますか？scanf の %s が一行の入力ではなく分かち書きされた単語だけを取ってくることに気がつきましたか？）

□ 課題 1.

同様に scanf で入力された文字列のうち、英字（a-z, A-Z）、数字、それ以外を個別にカウントするプログラムを作ってください。

考え方：
・英字の判定は ‘a’ 以上 ‘z’ 以下あるいは ‘A’ 以上 ‘Z’ 以下で良いか
・標準文字処理関数（教科書 p.162 表 9.4）に英字や数字判定関数もある

□ 標準文字列処理関数

★教科書 p. 183, 10. 4 「標準文字列処理関数」参照
C 言語は文字列を処理する機能をほとんど言語自体にもっていません。その代わり一般的な文字列処理を実現する関数が幾つも用意されています。以下に `char str1[10], str2[10];` などとして定義された文字配列変数 `str1, str2` を用いた例を示します。

・**strcpy :** 文字列を複写する（文字配列に文字列（あるいは別の文字配列）の中身をコピーする）
```
strcpy(str1, “sample”);
strcpy(str2, str1);
```
とすると、まず `str1` の内容が `“sample”` になり、次に `str2` の内容が `str1` に複写されます。つまり `str2` も最終的に `”sample”`になります。複写元（`str1`）の内容は変化しません。

・**strlen :** 文字列の長さ（文字数）を数える
```
printf(“%d\n”, strlen(str1) );
```
`str1` に`“sample”`が格納されていたとすると、`6` と出力されます。(配列のサイズ `10` ではなく、中の文字列の長さを数えます。また、終端文字がカウントされないことに注意してください。)

・**strcat :** 文字列を連結する
```
strcpy(str1, ”test”); strcpy(str2, ”sample”); strcat(str1, str2);
```
とすると `str2` の内容が `str1` に連結（追加）されます（`str1` が`”testsample”`になります)。`str2` の内容は変化しません。

・**strcmp :** 文字列を比較する
```
if(strcmp(str1, str2)==0) { .... } else { .... };
```
二つの文字列が同一であれば `0`、そうでなければ `0` 以外を返します。上記のようにして使います。

これらのライブラリ関数を利用するためには string.h のインクルードが必要です。
他にも `strncpy, strncat, strncmp` などがあります。興味があれば調べると良いでしょう。

□ 課題 2.

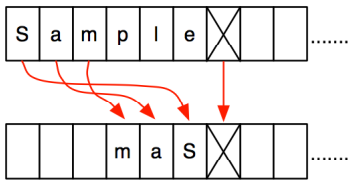
右のサンプルは strlen 関数を利用して文字数を数えながら、一文字ずつ縦に出力するものです。
これを修正して、入力された文字列を左右逆順にするプログラムを作成してください。例えば「Sample」と入力すると「elpmaS」という文字列を生成するのです。

```
char string[100];
int i;

printf("string >> ");
scanf("%s", string);

for(i=0; i<strlen(string); i++) {
  printf("#%c#\n", string[i]);
}
```

考え方：
もう一つの文字列変数（文字配列）を用意して、一文字ずつ移動させれば良いでしょう。添字 0 の文字は添字 5 の位置に、添字 1 の文字は添字 4 の位置に移動すれば良いわけです。最後にできあがった文字列を printf で出力してください。

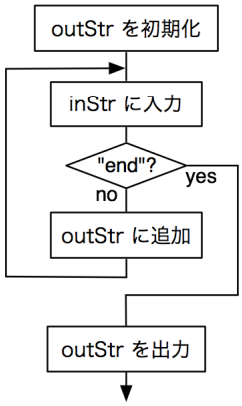


（進度次第では次回へ）

□ 課題 3.

入力した文字列を連結して、最後にまとめて連結した文字列として表示するプログラムを作ってください。end と入力されたら終了することにします。例えば「hello」「dear」「my」「friend」「end」と入力すると「hellodearmyfriend」と表示します。

処理の流れを右図に示しておきます。
・蓄積・出力用文字列変数（右例 outStr）は“”（空文字）で初期化
・文字列の追加は strcat 関数が使えるでしょう



□ 課題 4.

課題 3. を改良して、結果が「hello dear my friend」と単語ごとに区切られて表示されるようにしてください。また、まだ end と入力されなくても、結果が 20 バイトに届くようになったらその時点で出力してしまってください。
（画面の端まで行かずに折り返して表示するプログラムを作っていると思えばいいでしょう。）

右図の出力（# で囲まれた行）がに注目してください。「hello」「deear」「my」まで連結されたところで、次の「friend」を加えると 20 バイトを超えてしまうので、まず「hello dear my」までを出力し、「friend」は次の出力の先頭になっています。

```
$ ./string6
string >> hello
string >> dear
string >> my
string >> friend,
# hello dear my#
string >> welcome
string >> back.
#friend, welcome#
string >> end
#back.#
$
```

課題 3. でやったように、まずデータを処理する手順を検討し（必要であれば流れ図を描き）、それからプログラム修正に取りかかると良いでしょう。