2013 年 4 月 18 日（木）実施

標準入出力と変数

標準出力

前回の教材で扱った printf は，標準出力すなわち画面に，書式付き文字列をその書式に従って表示するためのライブラリ関数である。今回は，この書式の基本について説明する。

printf では，一対の二重引用符（"）で囲まれた文字列が画面に表示される。ただし，その文字列の中に，%で始まる文字列が含まれる場合には，**変換指定**という書式を表す文字列として扱われる。変換指定は%で始まり，**変換指定子**で終わる。一般的な変換指定の書き方は次のようになる。

% [フラグ] [最小フィールド幅] [精度] [長さ修飾子] 変換指定子

ここで，[ ] の部分はそれぞれ省略可能であり，最も単純なものは，%変換指定子 となる。この変換指定を printf で用いるには，次のように文字列の後に**式**を置く。

printf("x の値は%変換指定子である。¥n", 式);

この位置に表示　　式の値を評価

この式には，定数（値が一定のデータ），変数名，定数及び変数に演算を行うもの等が含まれる。変数は後述するが，データの格納場所である。printf の主な変換指定子の意味を次に挙げる。

変換指定子	意味
d	式の値を符号付き 10 進表記に変換する。
f	式の値を[-]dddd.dddddd の形の 10 進表現に変換する。
e	式の値を指数形式の 10 進表現に変換する。
c	式の値を符号なし文字型に変換して，その結果に対応する文字を出力する。
s	式の値を文字列が格納されているアドレスと捉え，その位置から終端の NULL バイト（'¥0'）が出てくるまでの文字列を出力する（終端文字は出力されない）。
x	式の値を符号なし 16 進表記(小文字)に変換する。

標準入力

scanf というライブラリ関数は，プログラム実行時に，標準入力すなわちキーボードからデータを受け取り，書式中の変換指定に従ってデータを変換し，指定された格納場所に書き込むために用いられる。一般的な変換指定の書き方は次のようになる。

% [代入抑止文字] [最大フィールド幅] [長さ修飾子] 変換指定子

ここで，[ ]の部分はそれぞれ省略可能であり，最も単純なものは，%変換指定子 となる。scanf の主な変換指定子の意味を次に挙げる。

変換指定子	意味
d	符号付きの 10 進の整数に対応する。
f	符号つき浮動小数点実数に対応する。
c	文字に対応する。
s	ホワイトスペース（スペース，タブ，改行等）ではない文字で構成された文字列に対応する。

変数

プログラム中の命令を通じて，メインメモリ上にデータを格納する領域を確保し，必要に応じてその場所に格納されるデータを上書きすることが可能である。このような領域を変数という。C 言語では，変数の取り扱いに関して，次のような特徴がある。

1. プログラム中で用いる変数は必ず**宣言**しておく。 ⇒ メインメモリ上にデータを格納する領域を確保せよ，という命令に相当する。

2. 変数の宣言時には，**データ型**を指定する。 ⇒ データ型毎にデータを格納する領域のサイズが固定されている。（サイズは処理系に依存する）

   例） int x;　/* 整数（integer）のデータ型の変数 x を宣言 */

3. 変数にデータを格納するには，**代入**を行う。

   例） x=10;　/* 変数 x に定数 10 を代入（=の右辺を評価し，その値を左辺の変数に格納）
   なお，宣言時に初期化することも可能である。 ⇒ int x=4; */

4. 代入式の左辺以外に変数名を用いると**参照**が行われ，変数に格納されたデータが取り出されて利用される。

   例） printf("x の値は%d である。¥n", x);　/* 変数 x の中身を%d の位置に表示 */

例題 1（**変数への代入**）

次のソースプログラムをテキストエディタで入力して，prog2-1.c の名前を付けて保存する。それを翻訳・編集して実行形式のファイルを作成し，実行せよ。

```
/* prog2-1.c */
#include <stdio.h>

int main(void)
```

```
{
    int x;

    x=10;
    printf("最初の代入の結果，現在の x の値は%d です。¥n", x);

    x=5;
    printf("次の代入の結果，現在の x の値は%d です。¥n", x);

    return 0;
}
```

例題２（**標準入力から取り込んだ整数データの変数への格納**）

次のソースプログラムをテキストエディタで入力して，prog2-2.c の名前を付けて保存する。それを翻訳・編集して実行形式のファイルを作成し，実行せよ。

```
/* prog2-2.c */
#include <stdio.h>

int main(void)
{
    int x;

    printf("x の値を整数で入力してください： ");
    scanf("%d", &x);

    printf("入力された x の値は%d です。¥n", x);

    return 0;
}
```

【解説】

1. 最初の printf を**ガイドコメント**と呼び，プログラムが実行される際に入力すべき内容を表示するためのものである，これが無いと画面には何も表示されずに入力待ちの状態となっている。

2. &x は変数 x のメインメモリ上の領域の先頭アドレスを表す。

例題３（**標準入力から取り込んだ文字データの変数への格納**）

次のソースプログラムをテキストエディタで入力して，prog2-3.c の名前を付けて保存する。それを翻訳・編集して実行形式のファイルを作成し，実行せよ。

なお，実行は複数回行い，その都度，別の文字を入力すること。特に，英字の大文字と小文字とが別の文字コードを有することを確認せよ。

```
/* prog2-3.c */
#include <stdio.h>

int main(void)
{
    char moji;

    printf("1 文字だけ入力してください： ");
    scanf("%c", &moji);

    printf("入力された文字は，[%c]です。¥n", moji);
    printf("その文字コードは，[%02x]です。¥n", moji);

    return 0;
}
```

【解説】

1. 変数 moji には，1 バイトコードの文字 1 文字分のデータを格納できる。

2. %02x の 0 はフラグの一種で，空白を 0 で埋めることを表す。それに続く 2 は最小フィールド幅で，結果を少なくとも 2 桁は表示することを表す。

演習１

次のソースプログラムの空欄 1)　　，2)　　を埋めて完成したものをテキストエディタで入力して，ex2-1.c の名前を付けて保存する。それを翻訳・編集して実行形式のファイルを作成し，実行せよ。

なお，x+y，x-y，x*y，x/y は，2 個の変数の間で加減乗除の演算を行った結果を表す。また，int x, y;により整数型の変数 x 及び y を宣言することができる。

```
/* ex2-1.c */
#include <stdio.h>

int main(void)
{
    int x, y;

    printf("x の値を整数で入力してください： ");
    scanf("%d", 1)    );

    printf("y の値を整数で入力してください： ");
    scanf("%d", 2)    );

    printf("x と y との和は%d です。¥n", x+y);
    printf("x と y との差は%d です。¥n", x-y);
    printf("x と y との積は%d です。¥n", x*y);
    printf("x と y との商は%d です。¥n", x/y);
```

```
    return 0;
}
```

提出物：例題 1，2，3 の出力結果をコピーして貼り付けたテキストファイル res2.txt (テキストエディタで作成)及び ex2-1.c の完成版のファイル