

2006年10月12日(木)実施

## 標準入出力と変数・配列

### 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、一對の二重引用符 (") 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.ddddd 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)。

### 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へ書き込むために用いられる。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。
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 */`

## 配列

メインメモリ上にデータを格納する領域を連続的に確保し、それらの領域を番号付けして扱うものを**配列**という。配列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配列は必ず**宣言**しておく。
2. 配列の宣言時には、**データ型及び要素数**を指定する。

例) `char str[11]; /* 文字 (character) のデータ型の配列 str を宣言 (要素数は 11, 扱える文字数は 1 バイトコードの文字の場合, 10 文字と終端となる。) */`

3. 配列の要素は 0 番から [要素数]-1 番までの添え字を用いて表され、各要素は変数と同様に扱える。⇒ 代入、参照は要素毎に行う。

例) `str[0]='a'; /* 配列 str の 0 番の要素に文字 'a' を代入 (C 言語では 1 文字は引用符 (')) で囲んで表し、文字列とは別の扱いとなる。) なお、宣言時に初期化することも可能である。⇒ char str[5]="abcd"; /*`

4. 配列名は、0 番の要素がメインメモリ上に占める領域の先頭アドレスを表す。

## 例題 1

次のプログラムを入力し、翻訳・編集して実行形式のファイルを作成し、実行せよ。ここで、ソースプログラム名は prog2-1.c とする。

```
/* 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。Yn", x);

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

return 0;
}
```

### 例題 2

次のプログラムを入力し, 翻訳・編集して実行形式のファイルを作成し, 実行せよ。ここで, ソースプログラム名は prog2-2.c とする。

```
/* 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。Yn", 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

次のプログラムを入力し, 翻訳・編集して実行形式のファイルを作成し, 実行せよ。ここで, ソースプログラム名は prog2-3.c とする。

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

int main(void)
{
    char str[21];

    printf("お名前を入力してください: ");
    scanf("%s", str);

    printf("今日は, %s さん!Yn", str);

    return 0;
}
```

**【解説】**

1. 配列 `str` には、1 バイトコードの文字であれば 20 文字分、2 バイトコードの文字であれば 10 文字分の文字を格納できる。
2. 配列名 `str` は、配列の 0 番の要素 `str[0]` が格納されているメインメモリ上の領域の先頭アドレスを表す。

**演習 1**

例題 3 のプログラムを実行する際に、「神保 雅人」のように途中で 1 バイトコードの空白文字を入れた場合の動作を確認せよ。また、「神保 雅人」のように途中で 2 バイトコードの空白文字を入れた場合の動作を確認せよ。