基礎プログラミング演習 II 教材 (#4)

■ 配列

（教科書 p.180「配列の基本」～p.183「配列の宣言」を参照）

□ 何故配列を使うか

右に 5 つのデータを入力し、出力するだけのプログラム（の断片）、その実行結果、そして配列をもちいた等価な働きをするプログラムを示します。

以下の点に注意してください。

- 三つのプログラムは全て同じ結果を出力する
- 5 つの記憶領域が必要だ
- 配列を使わず、5 つの変数で記憶領域を個別に取ると、5 回繰り返して（ほぼ同じ）コードを書く事になる（プログラム 1）
- 配列を使って 5 つぶんの記憶領域を簡単に用意できたが、それだけではやはり同じく繰り返しコードになる（プログラム 2）
- 配列とループを使った繰り返し処理によってコードを一回だけ書けば良いようにした（プログラム 3）
- ループを回るごとに「コードは同じ」だが、「対象となるデータ（変数）を変える」必要があるが、それは添字を変化させて実現した。
- 配列要素は 5 つだが、添字（変数 i）は 0～4 まで変化する。(1～5 ではない。そのために printf() 処理では i+1 している。)

> プログラム 3 では入力処理と出力処理を一つのループにまとめても良さそうに思えるだろう。しかし今回は入力処理と出力処理の間にデータ加工の仕事が行われることを想定して両者を分けている。(普通何かデータ処理するよね)

データを入力・保持して出力するだけのプログラム1

```
int data1, data2, data3, data4, data5;   ← 宣言

scanf("%d", &data1);
scanf("%d", &data2);
scanf("%d", &data3);                     } 入力処理
scanf("%d", &data4);
scanf("%d", &data5);

printf("data1 : %d\n", data1);
printf("data2 : %d\n", data2);
printf("data3 : %d\n", data3);           } 出力処理
printf("data4 : %d\n", data4);
printf("data5 : %d\n", data5);
```

プログラム1 の実行結果

```
$ echo 11 12 13 14 15 | ./a.out
data1 : 11
data2 : 12
data3 : 13                               } 出力結果
data4 : 14
data5 : 15
$
```

配列を用いて書いた等価な働きをするプログラム2

```
int data[5];   ← 宣言

scanf("%d", &data[0]);
scanf("%d", &data[1]);
scanf("%d", &data[2]);                   } 入力処理
scanf("%d", &data[3]);
scanf("%d", &data[4]);

printf("data1 : %d\n", data[0]);
printf("data2 : %d\n", data[1]);
printf("data3 : %d\n", data[2]);         } 出力処理
printf("data4 : %d\n", data[3]);
printf("data5 : %d\n", data[4]);
```

配列とループを用いて書いた等価な働きをするプログラム3

```
int i;
int data[5];   ← 宣言

for (i=0; i<5; i++) {
     scanf("%d", &data[i]);              } 入力処理
}

for (i=0; i<5; i++) {
     printf("data%d : %d\n", i+1, data[i]);   } 出力処理
}
```

つまり「同じコード」を「繰り返して利用」するが、その時「異なるデータ」を対象としていることが重要です。対象データの指定には「添字（インデックス）」を用います。

□ 文法的復習

教科書 p.190 「配列の記述のしかた」参照。

- 配列の初期化
- マクロ定数の利用（上のサンプルプログラムでは数字「5」が各所に登場していますね）

■ 練習（復習）

□ 課題 1. 逆順に出す

プログラム 3. を改造して、5 つの数を入力して、逆順に出力するようにしてください。
右に出力されている data[ ] の添字に注意してください。配列の中身は入力した順にデータが入ったまま、表示する順番を 5〜1 番目にするのです。

```
$ echo 11 12 13 14 15 | ./a.out
data[4] : 15
data[3] : 14
data[2] : 13
data[1] : 12
data[0] : 11
$
```

□ 課題 2. 最大の値を調べる

まず 5 つの数を入力して配列に保存し、次にその内容を調べて最も大きな数を出力してください。(一旦全部入力して配列に残し、後で配列の要素を頭から検査して最大のものを探す)
ただし入力する値はゼロ以上とします。

```
$  echo 22 80 57 60 50 | ./a.out
max : 80
$
```

網羅的なテストをすることを忘れないでください。つまり最大の数が入力の先頭にあったとき、最後にあったとき、複数あったとき、全てが同じ数だったとき、でも正しく動作することを検証してください。

□ 課題 3. 最大の値が幾つあったかを調べる

最大の数が複数あった場合に、それが幾つだったかを合わせて表示させて下さい。(右の出力例の赤文字箇所に注目)
ただしプログラム作成に当たって、以下の点に注意して下さい。

- いきなりコンピュータに向かってはいけません。あなたの思考力は半分以下になります。
- まずどのような変数を用意すれば良く、
- どのような処理手順で実現できるか明確にして、

その後で作業を始めるように。

```
$ echo 22 80 57 60 50 | ./a.out
max : 80 ( 1 個)
$ echo 22 80 57 80 50 | ./a.out
max : 80 ( 2 個)
$ echo 80 80 80 80 80 | ./a.out
max : 80 ( 5 個)
$
```

□ 課題 4. 最大の値とそれ以外を分けて出力する

右のように、最大の値と、それ以外の値を分けて出力してください。(最大値が二つあった場合の処置は右例に倣う)

```
$  echo 22 80 57 60 50 | ./a.out
max : 80
others :  22 57 60 50
$ echo 22 80 57 80 50 | ./a.out
max : 80
others :  22 57 50
$
```

□ 課題 5. サッと出来た人向けの提案

ここまでサッと出来た人は、以下のような点で工夫を加えてみてはどうでしょう。

課題 2.以降：
入力値の制限を外す、つまり負の数でも正しく処理できるようにする

課題 5.：
「全てが同じ数」の場合に分かりやすい（特別な）表示をさせる。恐らく普通に作ると others : だけが表示されるはずだが、これを others : (none) などとする。
あるいは「全ての入力は 80 でした」と出しても良い。

□ 課題 6. 二つめの配列に移す

プログラム 3. を改造して、5 つの要素を持つ配列をもう一つ用意し、そこに入力した 5 つの数を逆順にコピーして、その順で出力してください。
プログラムとしては右のようになるでしょう。
配列変数 data2[ ]が追加されている点に注意してください。
マクロ定数 NUM には 5 を定義しています。

```
int data1[NUM], data2[NUM];

for (i=0; i<NUM; i++) {
    scanf("%d", &data1[i]);
}

for (i=0; i<NUM; i++) {
    data2 に data1 の要素を移す処理（但し逆順に並べて）
}

for (i=0; i<NUM; i++) {
    printf("data2[%d] : %d\n", i, data2[i]);
}
```

右に出力されている data2[ ] の添字に注意してください。
添字は 0〜4 へと変化していますが、内容は入力の逆順になっています。

```
$ echo 11 12 13 14 15 | ./a.out
data2[0] : 15
data2[1] : 14
data2[2] : 13
data2[3] : 12
data2[4] : 11
$
```

□ 課題 7. 最大値以外のものだけを移す

課題 6.を改造して、入力した値の中で最大値を発見し、最大値以外の値だけを別の配列に(逆順に)移して下さい。課題 2.同様に、入力値はゼロ以上になるものとします。

これも良く考えてからコーディングしないと混乱すると思います。構えて作業してください。

```
$ echo 22 80 57 60 50 | ./a.out
max : 80
data2[0] : 50
data2[1] : 60
data2[2] : 57
data2[3] : 22
$ echo 22 80 57 80 50 | ./a.out
max : 80
data2[0] : 50
data2[1] : 57
data2[2] : 22
$
```

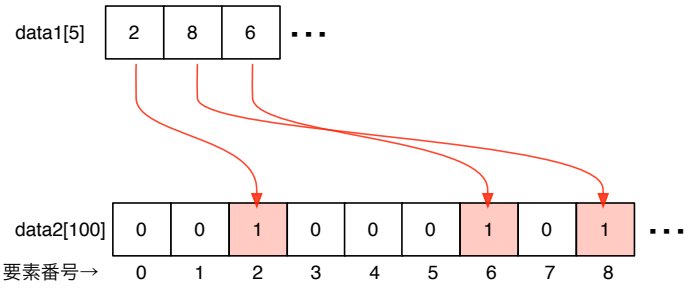
□ 課題 8. ソート（並べ替え）

配列を使って、最も簡単なソートを実現します。
入力する数値は 0〜99 までのランダムな値から 5 つ選んだものであり、重複はないもの、例えば 22, 80, 57, 60, 50 とします。これを右に示したように 22, 50, 57, 60, 80 の順に並べ直して表示させてください。

```
$ echo 22 80 57 60 50 | ./a.out
22 50 57 60 80
$
```

以下のアルゴリズムで実装してください。

- 配列を 100 個（0-99 まで）用意する
- 入力された値に対応する要素に、その値が登場したことをマークする（記録する）
- 後で配列要素を前から順に当たって、マークされている要素だけを出力する



□ 課題 9. ソート（重複あり対応）

課題 8.のアルゴリズムでは入力に重複があった場合、それらは一つにまとめられて出力されます。これを改善して、例えば右のように重複が合った場合にもそれらをまとめずに並べて出力するようにしてください。

```
$ echo 22 80 57 60 60 | ./a.out
22 57 60 60 80
$ echo 60 60 60 60 60 | ./a.out
60 60 60 60 60
$
```