ソフトゼミ∀　第６回 関数

■　発展: 配列引数

本編の方で書き忘れたのでこちらで解説。

変数同様、配列も引数にとることができる。しかし、若干振る舞いが普通の変数と異なるので注意が必要である。

ta06_1.c

```
#include <stdio.h>
void print_array( int a[], int n ){
   int i;
   for( i = 0; i < n; i++ ){
      printf( "%6d", a[ i ] );
   }
   printf("¥n");
}
void f( int a[], int b ){
   a[ 0 ] = 12345;
   b = 1;
}
int main( void ){
   int a[ 5 ] = { 75, 935, 1010, 310, 9999 };
   int b = 4;
   print_array( a, 5 );
   printf( "%d¥n", b );
   f( a, b );
   print_array( a, 5 );
   printf( "%d¥n", b );
   return 0;
}
```

print_array(int a[], int n)は、引数として与えられた配列の 0 要素目から n-1 要素目までを出力する関数である。このように、配列を受け取る時には「int (配列名)[]」だとか「int (配列名)[(要素数)]」だとか書く。(int *a という書き方もあるが、それは次回の∀あたりで解説したい。)

関数 f(int a[], int b)内では、引数として受け取った配列 a の a[0]に 12345 を代入、変数 b に 1 を代入している。

main 関数では、a, b を初期化した後に print_array で配列の中身を出力、変数 b の値を出力している。その後、関数 f を呼び出して、また print_array で配列の中身を出力、変数 b の値を出力している。配列が求められている引数に対しては配列名を実引数として渡してやればよい。

さて、出力結果はどうなったであろうか？多くの人は予想とは違う結果であっただろう。

2 回目の出力では a[0]の値が書き換わった(12345 になっていた)のに対して、b の値は 4 のまま変わっていない。なぜだろうか？

b の値がそのままなのは、本編でも解説した通りである。関数 f の方の b は、関数 f のローカル変数であり、関数 f の中でのみ有効である。それに対して、配列の方は…

これ以上の説明は次回の「ポインタ」の内容となってしまうので今回は解説しないが、今のところは「配列を引数として受け取った配列の各要素の値に変更があった場合、呼び出し元の関数内の配列の各要素の値も書き換わる」と覚えておくとよい。

■ 追加練習問題

1. 配列引数を用いて、フィボナッチ数列の各項を配列の各要素に格納する関数を作れ。
   ただし、フィボナッチ数列とは漸化式

$$\begin{cases} a_1 = 1 \\ a_2 = 1 \\ a_n = a_{n-2} + a_{n-1} \end{cases} \quad (n \geq 3, n \in \boldsymbol{N})$$

   の形であらわすことのできる数列である。
   a[0]には a_1 の値を…というように、第 n 項 a_n の値は a[n-1]に格納せよ。
   int 型でオーバーフローが発生する直前の項まで各項の値を求めること。
   配列 a はとりあえず 100 要素分くらい確保しておいてみるといいんじゃないかな？

2. 配列引数で受け取った int 型配列 a の a[0]～a[n - 1]を、昇順に並び替える関数 sort(int a[], int n)をつくれ。
   例えば、呼び出し元の関数内で、int a[5] = { 2, 7, 1, 8, 2 };の初期化直後に sort(a, 5);を呼ぶと、a の各要素は順番に 1, 2, 2, 7, 8 となる。