ソフトゼミA　第5回 解答と解説

■ 練習問題

1. 3人の身長[m]と体重[kg]を入力して、それぞれのBMIを計算し、一番やせている（BMIが低い）人の身長，体重を出力せよ。

```
#include <stdio.h>
struct SPerson{
   double height;
   double weight;
   double bmi;
};
int main( void ){
   struct SPerson person[ 3 ];
   int i, j;
   int min_idx = 0;
   for( i = 0; i < 3; i++ ){
      scanf( "%lf%lf", &person[ i ].height, &person[ i ].weight );
      person[ i ].bmi = person[ i ].weight / ( person[ i ].height * person[ i ].height );
      if( person[ i ].bmi < person[ min_idx ].bmi ){
         min_idx = i;
      }
   }
   printf( "height: %f [ m ], weight: %f [ kg ]¥n", person[ min_idx ].height, person[ min_idx ].weight );
    return 0;
}
```

構造体の配列です。int型などの普通の変数以外にも、構造体も配列にできます。この例では、main文の頭の方で、struct SPerson型の3要素の配列「person」を定義しています。このように、構造体を定義すると変数の型を作ることができます。

min_idxには、BMIの最小値の人の添字が入ります。このように、最小値or最大値を配列から探すときには、最小値or最大値が入っている要素の添字を取っておいて、比較の時にその番号の要素の値を引っ張ってくるのが良いでしょう。

ちなみに、配列・構造体無しでもできますが、配列・構造体を使う練習だと思ってください。

Sample Input
1.45 50.5 1.61 92.1 1.59 31.4
Sample Output
height: 1.590000 [m], weight: 31.400000[kg]

2. ある 5 人が大乱闘した時、撃墜数と自滅数 2 つの合計を求めるプログラムを作成せよ。入力の順番は 1 人目の撃墜数, 自滅数→ 2 人目の撃墜数, 自滅数 → … 5 人目の撃墜数, 自滅数である。

```
#include <stdio.h>
struct SPlayer{
   int defeatedNum;
   int ruinMyselfNum;
};
int main( void ){
   struct SPlayer player[ 5 ];
   int i;
   for( i = 0; i < 5; i++ ){
      scanf( "%d%d", &player[ i ].defeatedNum, &player[ i ].ruinMyselfNum );
   }
   for( i = 0; i < 5; i++ ){
      printf( "player %d: %2d¥n", i + 1, player[ i ].defeatedNum + player[ i ].ruinMyselfNum );
   }
   return 0;
}
```

ごめんなさい。いろいろ出題ミスしてしまいました＞＜

上の例では、ある人が撃墜した数と、その人が自滅した数の和をプレイヤーごとに表示しています。問題文の解釈によって、様々な解答が作れると思います。

Sample Input
3 1 4 1 5 9 2 6 5 3
Sample Output
player 1： 4
player 2： 5
player 3: 14
player 4: 8
player 5: 8

■ 追加練習問題

1. エラトステネスのふるいを用いて、100000 までの素数を全て求めよ。また、素数だけの配列を作り、935 番目の素数を求めよ。

```
#include <stdio.h>
int main( void ){
  int table[ 100001 ] = { 0 };
  int newTable[ 100001 ];
  int counter = 0;
  int i, j;
  table[ 1 ] = 1;
  for( i = 2; i <= 100000; i++ ){
    /* この continue; には、この回のループを終了し、
    (次の回があるなら)次の回のループを始める効果がある。*/
    if( table[ i ] ){ continue; }
    for( j = i * 2; j <= 100000; j += i ){
      table[ j ] = 1;
    }
  }
  for( i = 1; i <= 100000; i++ ){
    if( !table[ i ] ){
        printf( "%d¥n", i );
        table[ counter++ ] = i;
     }
  }
  printf( "The 935th prime number is %d.¥n", table[ 935 - 1 ] );
  return 0;
}
```

注目すべきは、配列の初期化 int table[100001] = { 0 };です。

ある数が素数であるかどうかを判定するために、1 から 100000 の添字を持った配列を作りたいですが、int table[100000]で手に入るのは、添字が 0 から 99999 までの配列です。100000 まで確かめる場合には、100001 要素分確保しておいて、その内の 1 から 100000 までの添字を使うようにすると使いやすいです。table[0]は使わないので、ある意味メモリを無駄にしていますが、人間にとってはわかりやすくなります。

この初期化式の右辺にある{ 0 }は、**配列の中身をすべて 0 で初期化する**特殊な初期化方法です。0 の場合のみ有効で、{1}とか{9999}というような初期化はできません。なおこの方法

を使わなくても、for(i = 1; i <= 100000; i++){ table[i] = 0; }という方法でももちろん OK です。

このサンプルの場合、配列の要素の中身が「0」のときチェックされていない状態、「1」の場合がチェックされている状態です。最後まで「0」(チェック無し)で残った整数が素数です。

最後に、素数を新しい配列に詰めていますが、このサンプルでは 0 から詰めているので、935 番目の素数は添字の 934 番に入っています。なお、table[counter++] = i;は、次の順番で処理が行われます。

I　table[counter] に i が代入される。(table の値は 1 増える前の値)
II　counter の値が 1 増える

もし、これを table[++ counter]と書くと、I と II が逆転します。注意しましょう。

935 番目の素数の値は、**7333** です。合っていましたか？

2.　(問題略)

```
#include <stdio.h>
int main( void ){
   int page[ 1001 ] = { 0 };
   int p, n, i;
   int temp;
   scanf( "%d%d", &p, &n );
   while( 1 ){
      scanf( "%d", &temp );
      if( temp == 0 ){ break; }
      page[ temp ]++;
   }
   for( i = 1; i <= p; i++ ){
      if( page[ i ] != n ){
         printf("%d ページ目: %3d 枚¥n", i, n - page[ i ] );
      }
   }
   return 0;
}
```

配列の初期化は、先程のエラトステネスのふるいの問題と同じ理由で int page[1001] = { 0 };としています。int 型配列 page には、見つかったページ番号と同

じ値の添字をもつ要素に、そのページが何枚見つかったかが入るようになっています。例えば、page[3]の値が 18 であれば、3 ページが 18 枚見つかったということです。

この調子で、足りないページ番号と、その枚数を全て出力すれば OK です。あとは、コードを読むなり改造するなりして理解しましょう。配列のこのような使い方を知っていると、結構便利ですよ！？