情報基礎 BII　課題の解説

柳本 豪一

ソースコード

課題 1

```
#include <stdio.h>

int main(void){
  int a, b, i, answer;

  printf("Input Number(1): ");
  scanf("%d", &a);
  printf("Input Number(2): ");
  scanf("%d", &b);

  answer = 1;
  for(i = 2; i <= a && i <= b; i++){
    if((a % i) == 0 && (b % i) == 0){
      answer = i;
    }
  }

  printf("Answer: %d %d\n", answer, a*b/answer);

  return(0);
}
```

課題 2

```
#include <stdio.h>

#define MAT1_ROW 2
#define MAT1_COL 2
#define MAT2_COL 2

void mul_mat(double a[][MAT1_COL],
  double b[][MAT2_COL], double c[][MAT2_COL]){
  int i, j, k;

  for(i = 0; i < MAT1_ROW; i++){
    for(j = 0; j < MAT2_COL; j++){
      c[i][j] = a[i][0] * b[0][j];
      for(k = 1; k < MAT1_COL; k++){
        c[i][j] += a[i][k] * b[k][j];
      }
    }
  }
}

int main(void){
  int i, j;
  double a[MAT1_ROW][MAT1_COL],
     [MAT1_COL][MAT2_COL], c[MAT1_ROW][MAT2_COL];

  for(i = 0; i < MAT1_ROW; i++){
    for(j = 0; j < MAT1_COL; j++){
      printf("Input A[%d][%d]: ", i, j);
      scanf("%lf", &a[i][j]);
    }
  }
  for(i = 0; i < MAT1_COL; i++){
    for(j = 0; j < MAT2_COL; j++){
      printf("Input B[%d][%d]: ", i, j);
      scanf("%lf", &b[i][j]);
    }
  }

  mul_mat(a, b, c);

  for(i = 0; i < MAT1_ROW; i++){
    for(j = 0; j < MAT2_COL; j++){
      printf("%lf ", c[i][j]);
    }
    printf("\n");
  }
```

```
  return(0);
}
```

課題 1

一番簡単な解法としては、最大公約数は 2 つの数字を両方割りきれる数字のなかで一番大きな数字であるので、これを探すものである。解答例では、上の解法を素直にプログラムにしています。ただ、注意しなくてはいけないのは割る数字をどこまで調べなくてはいけないかである。これは最大公約数を求める数字の小さい方まで調べればよい。したがって、この条件を満たすように for 文の条件判定式に利用している。最小公倍数は最大公約数が求まれば問題ないと思う。

他の解法としては、ユークリッドの互除法を用いる方法がある。こちらの方が探索回数が少なくなるので、高速な手法である。

課題 2

このプログラムでは、授業では詳しく触れなかったが 2 次元配列を関数の引数としてどのように渡すのか、関数で計算された行列の式をどうやって main 関数に戻すかが問題である。

まず、2 次元配列を関数の引数にすることについては授業のプレゼン資料のように関数を定義して呼び出せば問題がない。呼び出された関数内では、引数として渡された 2 次元配列は a[i][j] のように使える。

一方、行列の積の答えをどのように返すかであるが、return 文では一つの値しか返すことができないので、ここでは使えない。したがって、答えを保存する 2 次元配列を main 関数であらかじめ用意しておき、それを関数の引数として渡すことで答えを main 関数に戻すと言う方法をとっている。配列はポインタで受渡しされるので、関数側で配列に値を書き込むと main 関数側でもその変更が反映されるからである。

以下のような関数を作成して、ポインタを返すようなプログラムを作ったとしてもこれは正しくない。

```
double *mul_mat(double a[][MAT1_COL], double b[][MAT2_COL]){
   int i, j, k;
   double c[MAT1_ROW][MAT2_COL];

   for(i = 0; i < MAT1_ROW; i++){
      for(j = 0; j < MAT2_COL; j++){
         c[i][j] = a[i][0] * b[0][j];
         for(k = 0; k < MAT1_COL; k++){
            c[i][j] += a[i][k] * b[k][j];
         }
      }
   }

   return(c);
}
```

この理由は少し難しいが、2 次元配列 c は関数 mul_mat 内のローカル変数である。したがって、関数 mul_mat の処理が終わると消えてしまう運命にある。よって、関数の処理終了と同時に消えてしまう変数のアドレスを main 関数では受け取るので、希望の動作をしないこととなる。