lab07

```
$ gcc -DN=12 lab07.c

$ ./a.out < mat12.in
Matrix A is
 12 11 10  9  8  7  6  5  4  3  2  1
 11 12 11 10  9  8  7  6  5  4  3  2
 10 11 12 11 10  9  8  7  6  5  4  3
  9 10 11 12 11 10  9  8  7  6  5  4
  8  9 10 11 12 11 10  9  8  7  6  5
  7  8  9 10 11 12 11 10  9  8  7  6
  6  7  8  9 10 11 12 11 10  9  8  7
  5  6  7  8  9 10 11 12 11 10  9  8
  4  5  6  7  8  9 10 11 12 11 10  9
  3  4  5  6  7  8  9 10 11 12 11 10
  2  3  4  5  6  7  8  9 10 11 12 11
  1  2  3  4  5  6  7  8  9 10 11 12
det(A) = 13312
utime: 9.13577
```

```
score: 88.0
o. [Output] Program output is correct, good.
o. [Coding] lab07.c spelling errors: colunm(1), fuction(1), reduMtx(3)
o. [Format] Program format can be improved.
o. [Efficiency] can still be improved.
```

lab07.c

```
// EE231002 Lab07. Matrix Determinant
// 111060023, 黃柏霖
// 2022/11/04

#include <stdio.h>

#if !defined(N)
#define N 3
#endif

double det(double A[N][N], int dim);        // determine function declaration
What is this function for?

int main(void)
{
    int i, j;                               // loop control
    double A[N][N];                         // matrix


    for (i = 0; i < N; i++) {               // initialize the matrix
        for (j = 0; j < N; j++) {
            scanf("%lg", &A[i][j]);
        }
    }
    printf("Matrix A is\n");
    for (i = 0; i < N; i++) {               // print the matrix
        for (j = 0; j < N; j++) {
            printf("%3lg", A[i][j]);
        }
        printf("\n");
    }
    printf("det(A) = %lg\n", det(A, N));    // print the value of determinant
    return 0;
}

// This fuction is called det, the abbreviation for "determinant"
// It's purpose is to compute the determinant of a N * N matrix
// A[N][N] is the matrix, and dim is the range that should be computed
// det function return sum, the value of determinant of a dim * dim matrix
// No side effect
```

```
double det(double A[N][N], int dim)
{
    int col;                                // column of matrix
    int redurow;                            // row of reduced matrix
    int reducol;                            // colunm of reduced matrix
    double ReduMtx[N][N];                   // matrix whose order is reduced
    double sum = 0.0;                       // sum of each

    if (dim == 1)                           // if it's a 1 * 1 matrix
        sum = A[0][0];                      // return the only element
    else if (dim == 2)                      // if it's a 2 * 2 matrix
        sum = A[0][0] * A[1][1] - A[1][0] * A[0][1];    // compute directly
    else{
    else {
        for (col = 0; col < dim; col++) {   // the elements of first row
            // reduce the order of matrix for each elements
            for (redurow = 0; redurow < dim - 1; redurow++) {
                for (reducol = 0; reducol < dim - 1; reducol++) {
                    // For the reduMtx:
                    // the row will -1 since the first row is always eliminated
                    // the col will -1 if it's bigger than the col of element
                    if (reducol >= col) {
                        ReduMtx[redurow][reducol] = A[redurow + 1][reducol + 1];
                    }
                    else {
                        ReduMtx[redurow][reducol] = A[redurow + 1][reducol];
                    }
                }
            }
            // if element's col is 0, 2, 4..., than + element * det(reduMtx)
            // if element's col is 1, 3, 5..., than - element * det(reduMtx)
            if (col % 2 == 0)
                sum += A[0][col] * det(ReduMtx, dim - 1);
            else
                sum -= A[0][col] * det(ReduMtx, dim - 1);
        }
    }
    return sum;                             // return sum of determinant
}
```