

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define L 3
5  #define M 2
6  #define N 3
7
8  /* 行列の領域確保 */
9  double **dmatrix(int nr1, int nr2, int nl1, int nl2);
10 /* 行列の領域解放 */
11 void free_dmatrix(double **a, int nr1, int nr2, int nl1, int nl2);
12 /* 行列の積の計算 */
13 void matrix_product(double **a, double **b, double **c, int l1, int l2, int m1, int m2, int n1, int n2);
14
15
16 int main(void)
17 {
18     double **a, **b, **c;
19     int i, j;
20
21     a = dmatrix(1, L, 1, M); /* 行列 a[1...L][1...M] */
22     b = dmatrix(1, M, 1, N); /* 行列 b[1...M][1...N] */
23     c = dmatrix(1, L, 1, N); /* 行列 c[1...L][1...N] */
24
25     /* 行列 A の定義 */
26     for(i = 1; i <= L; i++){
27         for(j = 1; j <= M; j++){
28             a[i][j] = 2.0 * (i + j);
29         }
30     }
31     /* 行列 B の定義 */
32     for(i = 1; i <= M; i++){
33         for(j = 1; j <= N; j++){
34             b[i][j] = 2.0 * (i + j);
35         }
36     }
37
38     /* 行列の積の計算 */
39     matrix_product(a, b, c, 1, L, 1, M, 1, N);
40     /* 結果の表示 */
41     printf("A x B の結果は次の通りです. \n");
42     for(i = 1; i <= L; i++){
43         for(j = 1; j <= N; j++){
44             printf("%f ", c[i][j]);
45         }
46         printf("\n");
47     }
48
49     /* 行列領域の解放 */
50     free_dmatrix(a, 1, L, 1, M);
51     free_dmatrix(b, 1, M, 1, N);
52     free_dmatrix(c, 1, L, 1, N);
53
54     return 0;
55 }
56
57 double **dmatrix(int nr1, int nr2, int nl1, int nl2)
58 {
59     int i, nrow, ncol;
60     double **a;
61
62     nrow = nr2 - nr1 + 1; /* 行の数 */
63     ncol = nl2 - nl1 + 1; /* 列の数 */
64
65     /* 行の確保 */
66     if ((a = malloc(nrow * sizeof(double *))) == NULL)
67     {
68         printf("メモリが確保できません (行列 a)\n");
69         exit(1);
70     }
71     a = a - nr1; /* 行をずらす */
72
73     /* 列の確保 */
74     for (i = nr1; i <= nr2; i++)
75         a[i] = malloc(ncol * sizeof(double));
76     for (i = nr1; i <= nr2; i++)
77         a[i] = a[i] - nl1; /* 列をずらす */
78
79     return (a);
80 }
81
82 void free_dmatrix(double **a, int nr1, int nr2, int nl1, int nl2)
83 {
84     int i;
85
86     /* メモリの解放 */
87     for (i = nr1; i <= nr2; i++)
88         free((void *) (a[i] + nl1));
89     free((void *) (a + nr1));
90 }
91
92 void matrix_product(double **a, double **b, double **c, int l1, int l2, int m1, int m2, int n1, int n2)
93 {
94     int i, j, k;
95
96     for(i = l1; i <= l2; i++){ /* 行列の添え字 */
97         for(j = n1; j <= n2; j++){
98             c[i][j] = 0.0; /* 変数の初期化 */
99             for(k = m1; k <= m2; k++){ /* 列の添字 */
100                 c[i][j] += a[i][k] * b[k][j];
101             }
102         }
103     }
104 }

```