

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4
5 /* 関数の定義 */
6 double func1(double x);
7 double func2(double x);
8 /* ベクトル領域の確保 */
9 double *dvector(int i, int j);
10 /* ベクトル領域の解放 */
11 void free_dvector(double *a, int i);
12 /* ロンバーグ法 */
13 double romberg( double a, double b, int N, double eps, double (*f)(double) );
14
15 int main(void)
16 {
17     int N = 6;
18     double eps = pow(10.0, -10.0);
19
20     printf("2.0/(x*x) を [1,2] で積分します。最大反復回数は%dです\n", N);
21     printf("結果は%20.15fです\n", romberg(1.0, 2.0, N, eps, func1) );
22
23     printf("4.0/(1+x*x) を [0,1] で積分します。最大反復回数は%dです\n", N);
24     printf("結果は%20.15fです\n", romberg(0.0, 1.0, N, eps, func2) );
25
26     return 0;
27 }
28
29 /* ロンバーグ法 */
30 double romberg( double a, double b, int N, double eps, double (*f)(double) )
31 {
32     double S, h, *t, f0, f1;
33     int j, k, m, n;
34
35     t = dvector( 0, N );
36     h = b - a;
37     f0 = (*f)(a); f1 = (*f)(b);
38     t[0] = h*( f0 + f1 )/2.0;
39
40     /* ロンバーグ法 */
41     for( n = 1; n <= N; n++)
42     {
43         h = h / 2.0; S = 0.0;
44         for( j = 1; j <= (int)(pow(2.0,n)-1.0); j++ ) S += (*f)( a + j*h );
45         t[n] = h*( f0 + 2.0*S + f1 )/2.0;
46         if( fabs(t[n]-t[n-1]) < eps ) return t[n];
47         k = n;
48         for( m = 1; m <= n; m++)
49         {
50             k = k-1;
51             t[k] = ( pow(4.0,m)*t[k+1]-t[k] )/( pow(4.0,m)-1.0 );
52             if( k >= 1 && fabs(t[k]-t[k-1]) < eps ) return t[k];
53         }
54     }
55
56     /* ベクトル領域の解放 */
57     free_dvector(t,0);
58
59     return t[N]; /* 収束しなければ t[N] を積分値とする */
60 }
61
62 /* 関数の定義 */
63 double func1(double x)
64 {
65     return( 2.0/(x*x) );
66 }
67
68 double func2(double x)
69 {
70     return( 4.0 / (1.0+x*x) );
71 }
72
73
74 double *dvector(int i, int j) /* a[i]~a[j] の領域を確保 */
75 {
76     double *a;
77
78     if((a = malloc( ((j - i + 1) * sizeof(double))) ) == NULL)
79     {
80         printf("メモリが確保できません (from dvector) \n");
81         exit(1);
82     }
83
84     return (a - i);
85 }
86
87 void free_dvector(double *a, int i)
88 {
89     free( (void *) (a + i) ); /* (void *) 型へのキャストが必要 */
90 }

```