

```

1 #include <stdio.h>
2 #include <math.h>
3 #include <stdlib.h>
4
5 double *dvector(long i, long j); /* ベクトル領域の確保 */
6 void free_dvector(double *a, long i); /* 領域の解放 */
7 double func(double x, double y); /* 関数の定義 */
8 /* ルンゲ・クッタ法 */
9 double *rk4( double y0, double *y, double a, double b, int n,
10              double (*f)(double, double) );
11 /* アダムス法 */
12 double *adams( double y0, double *y, double a, double b, int n,
13               int N, double eps, double (*f)(double, double) );
14
15 int main(void)
16 {
17     double *y, h, a=0.0, b=1.0, y0=1.0, eps=pow(10.0,-8.0) ;
18     int i, n, N=10; /* 最大反復回数 N */
19
20     printf("分割数を入力してください--->");
21     scanf("%d",&n);
22
23     y = dvector( 0, n ); /* 領域の確保 */
24
25     /* アダムス法 */
26     y = adams( y0, y, a, b, n, N, eps, func );
27
28     /* 結果の表示 */
29     h = (b-a)/n ; /* 刻み幅 */
30     for ( i = 0 ; i <= n ; i++)
31     {
32         printf("x=%f \t y=%f \n", a+i*h, y[i] );
33     }
34
35     free_dvector( y, 0 );
36     return 0;
37 }
38
39 /* アダムス法 */
40 double *adams( double y0, double *y, double a, double b, int n,
41               int N, double eps, double (*f)(double, double) )
42 {
43     double yp, h, *F, x;
44     int i, j;
45
46     y = dvector( 0, n ); /* y[0,1,...n] の確保 */
47     F = dvector( 0, 4 ); /* F[0,1,...4] の確保 */
48     h = (b-a)/n; /* 刻み幅の設定 */
49
50     /* スタート */
51     y = rk4( y0, y, a, b, n, f);
52     x = a;
53     for ( i = 0; i <= 3; i++)
54     {
55         F[i] = f(x,y[i]); x += h;
56     }
57
58     /* 反復計算 */
59     for ( i = 3; i <= n-1; i++)
60     {
61         /* アダムス・バッシュフォース法 */
62         F[3] = f(x-h, y[i]);
63         yp = y[i] + h*(55.0*F[3]-59.0*F[2]+37.0*F[1]-9.0*F[0])/24.0;
64         for ( j = 1; j <= N; j++ )
65         {
66             /* アダムス・ムルトン法 */
67             F[4] = f(x,yp);
68             y[i+1] = y[i] + h*(9.0*F[4]+19.0*F[3]-5.0*F[2]+F[1])/24.0;
69             if ( fabs(y[i+1]-yp) < eps ) break;
70             yp = y[i+1];
71         }
72         for ( j = 1; j <= 4; j++) F[j-1] = F[j]; /* F[i] の更新 */
73         x += h;
74     }
75
76     free_dvector( F, 0 ); /* 領域の解放 */
77     return y;
78 }
79
80 /* ルンゲ・クッタ法 */
81 double *rk4(double y0, double *y, double a, double b, int n,
82             double (*f)(double, double))
83 {
84     double k1, k2, k3, k4, h, x;
85     int i;
86
87     h = (b - a) / n;
88     /* 初期値の設定 */
89     y[0] = y0;
90     x = a;
91
92     /* ルンゲ・クッタ法 */
93     for (i = 0; i < n; i++)
94     {
95         k1 = f(x, y[i]);
96         k2 = f(x + h / 2.0, y[i] + h * k1 / 2.0);
97         k3 = f(x + h / 2.0, y[i] + h * k2 / 2.0);
98         k4 = f(x + h, y[i] + h * k3);
99         y[i + 1] = y[i] + h / 6.0 * (k1 + 2.0 * k2 + 2.0 * k3 + k4);
100        x += h;
101    }
102
103    return y;
104 }
105
106 /* 関数の定義 */
107 double func(double x, double y)
108 {
109     return( x + y );
110 }
111
112 double *dvector(long i, long j) /* a[i]~a[j] の領域を確保 */
113 {
114     double *a;
115
116     if ((a = malloc(((j - i + 1) * sizeof(double)))) == NULL)
117     {
118         printf("メモリが確保できません (from dvector) \n");
119         exit(1);
120     }
121
122     return (a - i);
123 }
124
125 void free_dvector(double *a, long i)
126 {
127     free((void *) (a + i)); /* (void *) 型へのキャストが必要 */
128 }
129

```