```c
#include <stdio.h>
#include <stdlib.h>

double *dvector(long i, long j);            /* ベクトル領域の確保 */
void free_dvector(double *a, long i);       /* 領域の解放 */
void FUNC(double x, double *y, double *f);  /* 関数の定義 */
/* ルンゲ・クッタ法(N 変数版) */
void rk4_m( double *y, double *f, int N,
           double a, double b, int n, void (*F)(double, double *, double *) );

int main(void)
{
  double *y, *f, a=0.0, b=3.1415926535897932;
  int n, N=2;   /* N 変数 */

  y = dvector( 1, N ); f = dvector( 1, N );   /* 領域の確保 */
  y[1]=1.0;y[2]=0.0;   /* 初期値の設定 */

  printf("分割数を入力してください--->");
  scanf("%d",&n);

  rk4_m( y, f, N, a, b, n, FUNC );  /* ルンゲ・クッタ法 */

  return 0;
}

/* ルンゲ・クッタ法(N 変数版) */
void rk4_m( double *y, double *f, int N,
           double a, double b, int n, void (*F)(double, double *, double *) )
{
  double *k1, *k2, *k3, *k4, h, x, *tmp;
  int i, j;

  k1 = dvector( 1, N );   k2 = dvector( 1, N );
  k3 = dvector( 1, N );   k4 = dvector( 1, N );
  tmp = dvector( 1, N );

  /* 初期値の設定・表示 */
  h = (b-a)/n;          /* 刻み幅 */
  x = a;
  printf("x=%f \t y1=%f \t y2=%f \n", x, y[1], y[2]);

  /* ルンゲ・クッタ法(N 変数版) */
  for ( i = 0 ; i < n ; i++)
  {  /* k1 の計算 */
    (*FUNC)( x, y, f );
    for( j = 1; j <= N; j++ ) k1[j] = f[j];
    for( j = 1; j <= N; j++) tmp[j] = y[j] + h*k1[j]/2.0;
    /* k2 の計算 */
    (*FUNC)( x+h/2.0, tmp, f );
    for( j = 1; j <= N; j++ ) k2[j] = f[j];
    for( j = 1; j <= N; j++ ) tmp[j] = y[j] + h*k2[j]/2.0;
    /* k3 の計算 */
    (*FUNC)( x+h/2.0, tmp, f );
    for( j = 1; j <= N; j++ ) k3[j] = f[j];
    for( j = 1; j <= N; j++ ) tmp[j] = y[j] + h*k3[j];
    /* k4 の計算 */
    (*FUNC)( x+h, tmp, f );
    for( j = 1; j <= N; j++ ) k4[j] = f[j];
    for( j = 1; j <= N; j++ )
      y[j] = y[j] + h/6.0*( k1[j] + 2.0*k2[j] + 2.0*k3[j] + k4[j] );
    x += h;
    printf("x=%f \t y1=%f \t y2=%f \n", x, y[1], y[2]);
  }

  /* 領域の解放 */
  free_dvector( k1, 1 );  free_dvector( k2, 1 );
  free_dvector( k3, 1 );  free_dvector( k4, 1 );
  free_dvector( tmp, 1 );
}

/* 関数の定義 */
void FUNC(double x, double *y, double *f)
{
  f[1] = y[2];
  f[2] = -y[1];
}

double *dvector(long i, long j) /* a[i]～a[j] の領域を確保 */
{
  double *a;

  if ((a = malloc(((j - i + 1) * sizeof(double)))) == NULL)
  {
    printf("メモリが確保できません (from dvector) \n");
    exit(1);
  }

  return (a - i);
}

void free_dvector(double *a, long i)
{
  free((void *)(a + i)); /* (void *) 型へのキャストが必要 */
}
```