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

#define  M  6    /* データのべア数 */
#define  N  3    /* N 次式で近似 */

/* ベクトルの入力 */
void input_vector2( double *b, char c, int n, FILE *fin, FILE *fout);
/* 行列の領域確保 */
double **dmatrix(int nr1, int nr2, int nl1, int nl2);
/* 行列の領域解放 */
void free_dmatrix(double **a, int nr1, int nr2, int nl1, int nl2);
/* ベクトル領域の確保 */
double *dvector(int i, int j);
/* ベクトル領域の解放 */
void free_dvector(double *p, int i);
/* 部分ピボット選択付き ガウス消去法 */
double *gauss2( double **a, double *b, int n );
/* 最小2乗近似 */
void least_square( double *x, double *y, FILE *fout );


int main(void)
{
  FILE *fin, *fout;
  double *x, *y;

  /* ベクトルの領域確保 */
  x = dvector(1,M); /* x[1...M] */
  y = dvector(1,M); /* y[1...M] */

  /* ファイルのオープン */
  if ( (fin = fopen( "input_func.dat", "r")) == NULL )
    {
      printf("ファイルが見つかりません : input_func.dat \n");
      exit(1);
    }
  if( (fout = fopen( "output_func.dat", "w")) == NULL )
  {
    printf("ファイルが作成できません : output_func.dat \n");
    exit(1);
   }

  input_vector2( x, 'x', M, fin, fout );    /* ベクトル x の入出力 */
  input_vector2( y, 'y', M, fin, fout );    /* ベクトル y の入出力 */


  least_square( x, y, fout );  /* 最小2乗法 */

  fclose(fin); fclose(fout);  /* ファイルのクローズ */

  /* 領域の解放 */
  free_dvector( x, 1 );  free_dvector( y, 1 );

  return 0;
}

void least_square( double *x, double *y, FILE *fout )
{
  double *a, **p;
  int i, j, k;

  p = dmatrix(1, N+1, 1, N+1);  /* p[1...N+1][1...N+1] */
  a = dvector(1, N+1);          /* a[1...N+1] */

  /* 右辺ベクトルの作成 */
  for(i=1; i <= N+1; i++)
  {
    a[i]=0.0;
    for( j = 1; j <= M; j++)
    {
      a[i] +=  y[j]*pow(x[j],(double)(i-1)) ;
    }
  }

  /* 係数行列の作成 */
  for( i = 1; i <= N+1; i++)
  {
    for( j = 1; j <= i;  j++ )
    {
      p[i][j]=0.0;
      for( k =1; k <= M; k++)
      {
        p[i][j] += pow( x[k], (double)(i+j-2) );
      }
      p[j][i] = p[i][j];
    }
  }
  /* 連立一次方程式を解く．結果は a に上書き */
   a = gauss2( p, a, N+1 );

  /* 結果の出力 */
  fprintf( fout, "最小2乗近似式は y=\n");
  for( i = N+1 ; i >= 1 ; i--)
  {
    fprintf(fout, "+ %5.2f x^%d ", a[i],i-1);
  }
  fprintf(fout, "\n");

  /* 領域の解放 */
  free_dmatrix( p, 1, N+1, 1, N+1 ); free_dvector( a, 1 );
}

/* 部分ピボット選択付き ガウス消去法 */
double *gauss2( double **a, double *b, int n )
{
  int i, j, k, ip;
  double alpha, tmp;
  double amax, eps=pow(2.0, -50.0); /* eps = 2^(-50)とする */

  for( k = 1; k <= n-1; k++)
  {
    /* ピボットの選択 */
    amax = fabs(a[k][k]); ip = k;
    for( i = k+1; i <= n; i++)
    {
      if ( fabs(a[i][k]) > amax )
      {
        amax = fabs(a[i][k]); ip = i;
      }
    }
    /* 正則性のﾁｪｯｸ */
    if ( amax < eps ) printf("入力した行列は正則ではないﾔﾂﾈﾝ");
    /* 行交換 */
    if ( ip != k)
    {
      for( j = k; j <= n; j++)
      {
        tmp = a[k][j]; a[k][j]=a[ip][j]; a[ip][j]=tmp;
      }
      tmp = b[k] ; b[k]=b[ip]; b[ip]=tmp;
    }
    /* 約後消去 */
    for( i = k+1; i <= n; i++)
    {
      alpha = - a[i][k]/a[k][k];
      for( j = k+1; j <= n; j++)
      {
        a[i][j] = a[i][j] + alpha * a[k][j];
      }
      b[i] = b[i] + alpha * b[k];
    }
  }

  /* 後退代入 */
  b[n] = b[n]/a[n][n];
  for( k = n-1; k >= 1; k--)
  {
    tmp = b[k];
    for( j = k+1; j <= n; j++)
    {
      tmp = tmp - a[k][j] * b[j];
    }
    b[k] = tmp/a[k][k];
  }

  return b;
}

/* b[1...n] の入力 */
void input_vector2( double *b, char c, int n, FILE *fin, FILE *fout)
{
  int i;

  fprintf( fout, "ベクトル%c は次の通りです\n", c);
  for( i = 1 ; i <= n ; i++)
  {
    fscanf(fin, "%lf", &b[i]);
    fprintf( fout, "%5.2f\t", b[i]);
  }
  fprintf( fout, "\n");
}

double **dmatrix(int nr1, int nr2, int nl1, int nl2)
{
        int i, nrow, ncol;
        double **a;

        nrow = nr2 - nr1 + 1; /* 行の数 */
        ncol = nl2 - nl1 + 1; /* 列の数 */

        /* 行の確保 */
        if((a = malloc(nrow * sizeof(double *)) )== NULL){
                printf("メモリが確保できません (行列 a)\n");
                exit(1);
        }
        a = a - nr1; /* 行をずらす */

        /* 列の確保 */
        for(i = nr1; i <= nr2; i++) a[i] = malloc(ncol * sizeof(double));
        for(i = nr1; i <= nr2; i++) a[i] = a[i] - nl1;                  /* 列をずらす */

        return (a);
}

void free_dmatrix(double **a, int nr1, int nr2, int nl1, int nl2)
{
        int i;

        /* メモリの解放 */
        for(i = nr1; i <= nr2; i++) free((void *)(a[i] + nl1));
        free((void *)(a - nr1));
}

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

/* ベクトル a[m...n] と b[m...n] の内積を計算する */
double inner_product(int m, int n, double *a, double *b)
{
  int i;
  double s = 0.0;

  for (i = m; i <= n; i++)
    s += a[i] * b[i];

  return s;
}
```