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

#define N    10                 /* N元方程式 */
#define EPS  pow(10.0, -8.0)     /* epsilonの設定 */
#define KMAX 100                /* 最大反復回数 */

/* 行列の入力 */
void input_matrix( double **a, char c, FILE *fin, FILE *fout);
/* ベクトルの入力 */
void input_vector( double *b, char c, 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);
/* 領域の解放 */
void free_dvector(double *a, int i);
/* 1 ノルムの計算 a[m...n] */
double vector_norm1( double *a, int m, int n );
/* A 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);
/* 行列a[1...N][1...N] とベクトル b[1...N] との積 c<-Ab */
void matrix_vector_product( double **a, double *b, double *c );
/* 共役勾配法(CG法) */
double *cg( double **a, double *b, double *x );

int main(void)
{
  FILE *fin, *fout;
  double **a, *b, *x;
  int i;

  /* 行列およびベクトルの領域確保 */
  a = dmatrix(1, N, 1, N); /* 行列 a[1...N][1...N] */
  b = dvector(1,N); /* b[1...N] */
  x = dvector(1,N); /* x[1...N] */

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

  input_matrix( a, 'A', fin, fout );    /* 行列 A の入出力 */
  input_vector( b, 'b', fin, fout );    /* ベクトル b の入出力 */
  input_vector( x, 'x', fin, fout );    /* 初期ベクトル x0 の入出力 */
  x = cg( a, b, x );                    /* 共役勾配法(CG法) */

  /* 結果の出力 */
  fprintf( fout, "Ax=b の解は次の通りです\n");
  for( i = 1 ; i <= N ; i++ )
  {
    fprintf(fout, "%f\n", x[i]);
  }

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

  /* 領域の解放 */
  free_dmatrix( a, 1, N, 1, N ); free_dvector( b, 1 ); free_dvector( x, 1 );

  return 0;
}

/* 共役勾配法(CG法)  */
double *cg(double **a, double *b, double *x)
{
  double eps, *r, *p, *tmp, alpha, beta, work;
  int i, k=0;

  r = dvector(1,N); /* r[1...N] */
  p = dvector(1,N); /* p[1...N] */
  tmp = dvector(1,N); /* tmp[1...N] */

  matrix_vector_product( a, x, tmp );  /* tmp <- A b */

  for( i = 1; i <= N; i++)
  {
    p[i] = b[i] - tmp[i] ; r[i] = p[i];
  }

  do
  {
    /* alpha の計算 */
    matrix_vector_product( a, p, tmp );  /* tmp <- A p_k */
    work = inner_product( 1, N, p, tmp); /* work <- (p,Ap_k) */
    alpha = inner_product( 1, N, p, r ) / work ;

    /* x_{k+1} と r_{k+1}の計算 */
    for( i = 1; i <= N; i++) x[i] = x[i] + alpha*p[i];
    for( i = 1; i <= N; i++) r[i] = r[i] - alpha*tmp[i];

    /* 収束判定 */
    eps = vector_norm1(r, 1, N);
    k++;  /* 反復回数の更新 */
    if ( eps < EPS )  goto OUTPUT;

    /* beta と p_{k+1} の計算 */
    beta = - inner_product( 1, N, r, tmp) / work;
    for( i = 1; i <= N; i++) p[i] = r[i] + beta*p[i];
 }while( k < KMAX );

  OUTPUT:;
  /* 領域の解放 */
  free_dvector( r, 1 ); free_dvector( p, 1 ); free_dvector( tmp, 1 );
  if ( k == KMAX )
  {
    printf("答えが見つかりませんでした\n");
    exit(1);
  }
  else
  {
    printf("反復回数は%d回です\n", k); /* 反復回数を画面に表示 */
    return x;
  }
}

/* 行列 a[1...N][1...N] とベクトル b[1...N] との積 c <- Ab */
void matrix_vector_product(double **a, double *b , double *c)
{
  double wk;
  int i, j;

  for ( i = 1; i <= N; i++)
  {
    wk = 0.0;
    for ( j = 1; j <= N; j++ )
    {
      wk += a[i][j]*b[j];
    }
    c[i] = wk;
  }
}

/* a[1...N][1...N] の入力 */
void input_matrix(double **a, char c, FILE *fin, FILE *fout)
{
  int i, j;

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

/* b[1...N]の入力 */
void input_vector(double *b, char c, 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)
{
  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 */
}

/* 比較関数 (昇順) */
int double_comp(const void *s1, const void *s2)
{
  const double a1 = *((double *)s1); /* (double *) へキャスト */
  const double a2 = *((double *)s2); /* (double *) へキャスト */
  if (a1 < a2)
  {
    return -1;
  }
  else if (a1 == a2)
  {
    return 0;
  }
  else
  {
    return 1;
  }
}

/* 最大値ノルムの計算 a[m...n] */
double vector_norm_max(double *a, int m, int n)
{
  int i, tmp;
  tmp = n - m + 1; /* 全要素数の計算 */
  for (i = m; i <= n; i++)
    a[i] = fabs(a[i]);
  /* 並べ換え:先頭アドレスが m だけずれていることに注意 */
  qsort(a + m, tmp, sizeof(a[0]), double_comp);
  return a[n];
}

/* 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 ;
}

/* 1 ノルムの計算 a[m...n] */
double vector_norm1(double *a, int m, int n)
{
  int i;
  double norm = 0.0;
  for (i = m; i <= n; i++)
  {
    norm += fabs(a[i]);
  }
  return norm;
}
```