```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);
/* ベクトルの領域の確保 */
double *dvector(int i, int j);
/* 領域解放 */
void free_dvector(double *a, int i);
/* 比較関数 */
int double_comp( const void *s1 , const void *s2 );
/* 最大値/ノルムの計算 a[m...n] */
double vector_norm_max( double *a, int m, int n );
/* ガウス・ザイデル法 */
double *gauss_seidel(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); /* 行 b[1...N] */
    x = dvector(1,N); /* 行 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("ファイルが外力できません : output_sp.dat \n");
        exit(1);
    }

    input_matrix( a, 'A', fin, fout );    /* 行列 A の入力 */
    input_vector( b, 'b', fin, fout );    /* ベクトル b の入力 */
    input_vector( x, 'x', fin, fout );    /* 初期ベクトル x0 の入力 */
    x = gauss_seidel( a, b, x );          /* ガウス・ザイデル法 */

    /* 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;
}

/* ガウス・ザイデル法 */
double *gauss_seidel(double **a, double *b, double *x)
{
    double eps, *xo, s, t;
    int i, j, k=0;

    xo = dvector(1,N); /* xo[1...N] */

    do
    {
        /* xo <- x_k, x <- x_{k+1} */
        for( i = 1; i <= N; i++ ) xo[i] = x[i];    /* x_k に x_{k+1} を代入 */
        /* i+1 の処理 */
        t = 0.0;
        for( j = 2; j <= N; j++ ) t += a[1][j]*xo[j];
        x[1] = ( b[1] - t )/a[1][1];
        /* i=2,3,...N の処理 */
        for( i = 2; i <= N; i++ )
        {
            s = 0.0; t = 0.0;
            for( j = 1; j < i; j++ )  s += a[i][j]*x[j];      /* i-1 列までの和 */
            for( j = i+1; j <= N; j++ ) t += a[i][j]*xo[j];  /* i+1 列以降の和 */
            x[i] = ( b[i] - s - t )/a[i][i];
        }
        for( i = 1; i <= N; i++ )  xo[i] = xo[i]-x[i];
        eps = vector_norm_max(xo, 1, N);
        k++;
    }while(eps > EPS && k < KMAX);

    free_dvector( xo, 1 ); /* 領域の解放 */
    if ( k == KMAX )
    {
        printf("答え方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;
    }
}

/* 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]);
    /* 各々の要素に絶対値がとられていることに注意 */
    qsort(a + m, tmp, sizeof(a[0]), double_comp);
    return a[n];
}
```