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

enum
{
    FILES = 0,
    FORMULES = 1,
    WP = 1,
    DET = 1 << 1,
    DISCRP = 1 << 2,
    REV = 1 << 3,
    COND = 1 << 4,
};

int flags =  COND | DET | DISCRP | REV;

// данные для примера 2
const int M = 2, N = 40; // пункт 2.2 приложение 2
const double pi = 3.1415926535;

void check_print(double **a, double *x, int n);
void init_matr(FILE *f, double **a, double *x, int n, int mode);
void gauss_mod(double **a, double *x, double *f, int n);
double *mul_matr(double **a, double *x, int n);
double **copy_1(double **a, double **,  int n);
double *copy_2(double *f, double *, int n);
double **create_rev(double **rev, int n);
double norm(double **a, int n);
int max_replace(double **a, double *b, int n, int i, double **rev);

int main(int argc, char **argv)
{
    FILE *fin = fopen(argv[1], "r");
    int n;
    fscanf(fin, "%d", &n);

    int mode;
    sscanf(argv[2], "%d", &mode);

    if (mode == 1)
        n = N;

    double **A, *x, *f;
    x = calloc(n, sizeof(double));
    f = calloc(n, sizeof(double));
    A = calloc(n, sizeof(double *));



    init_matr(fin, A, f, n, mode);
//check_print(A, f, n);
    gauss_mod(A, x, f, n) ;
//check_print(A, x, n);


    return 0;
}

void gauss_mod(double **a, double *x, double *f, int n)
{
    int i, j, k , m, replace_cnt = 0;
    double first, determinant = 1. , cond_num = 0. , **a_cpy, **rev, *f_cpy;

    if (flags & DET)
        printf("DET - ON\n");

    if (flags & WP)
        printf("WP - ON\n");
```

```
70
71      if (flags & DISCRP) {
72          printf("DISCRP - ON\n");
73          a_cpy = copy_1(a_cpy, a, n);
74          f_cpy = copy_2(f_cpy, f, n);
75      }
76
77      if (flags & COND) {
78          printf("COND - ON\n");
79          cond_num += norm(a, n);
80      }
81
82      if ((flags & REV) || (flags & COND)) {
83          if (flags & REV)
84              printf("REV - ON\n");
85          rev = create_rev(rev, n);
86      }
87
88      printf("\n");
89
90      for (i = 0; i < n; i++) {
91          if (flags & WP) // fkag WP - it's gauss with a principal element
92              replace_cnt += max_replace(a, f, n, i, rev);
93
94          for (j = i; j < n; j++) { // делим строку на первый строки (приводим
    к единице первый эл)
95              if (a[j][i] == 0) first = 1;
96              else first = a[j][i];
97
98              for (k = 0; k < n; k++) // делим на старший элемент
99                  a[j][i+k] /= first;
100
101             if ((flags & REV) || (flags & COND))
102                 for (int v = 0; v < n; v++)
103                     rev[j][v] /= first;
104
105             f[j] /= first;
106
107             if (flags & DET) // паралелллльное вычисление определителя)
108                 determinant *= first;
109         }
110
111         for (m = i+1; m < n; m++){ // вычитаем строки друг из друга
112             if (a[m][i] == 0) continue;
113
114             for (k = 0; k < n-i; k++)
115                 a[m][k+i] -= a[i][i+k];
116
117             if ((flags & REV) || (flags & COND)) // работаетм с флагом для -1
    матрицы
118                 for(int v = 0; v < n; v++)
119                     rev[m][v] -= rev[i][v];
120
121             f[m] -= f[i]; // и столбец значений тоже вычитаем (для гауса)
122         }
123     }
124
125     for (int g = n-1; g >= 0; g--) { // обратный ход метода гауса
126         for(int h = g - 1; h >= 0; h--) {
127             f[h] -= f[g]*a[h][g];
128
129             if((flags & REV) || (flags & COND))
130                 for(int v = 0; v < n; v++)
131                     rev[h][v] -= rev[g][v] * a[h]
    [g];
132
133             a[h][g] -= a[g][g] * a[h][g]; // как будто бы вычли
134         }
135
```

```
136             x[g] = f[g];
137         }
138 
139 
140         printf("x: ");
141         for(int h = 0; h < n; h++)
142             printf("%lf ", x[h]);
143         printf("\n");
144 
145         determinant *= pow(-1., replace_cnt); // если была перестановка строк
146         if (flags & DET)
147             printf("Determinant = %lf\n", determinant);
148 
149         if (flags & DISCRP) { // считаем невязку
150             double *res = mul_matr(a_cpy, x, n);
151             double dis = 0.;
152             for (int h = 0; h < n; h++)
153                 dis += (f_cpy[h]-res[h]) * (f_cpy[h]-res[h]);
154             dis = sqrt(dis);
155             printf("Discrepancy = %lf\n", dis);
156         }
157 
158         if (flags & COND) {
159             cond_num += norm(rev, n);
160             printf("Condition number: %lf\n", cond_num);
161         }
162         if (flags & REV) {
163             printf("Revers:\n");
164             for(int g = 0; g < n; g++){
165                 for(int h = 0; h< n; h++){
166                     printf("%lf ", rev[g][h]);
167                 }
168                 printf("\n");
169             }
170         }
171 
172         return ;
173     }
174 
175     double *mul_matr(double **a, double *x, int n)
176     {
177         double *res = calloc(n, sizeof(double));
178         for (int i = 0; i < n; i++){
179             for(int j = 0; j < n; j++){
180                 res[i] += a[i][j] * x[j];
181             }
182         }
183         return res;
184     }
185 
186     int max_replace(double **a, double *b, int N, int i, double **rev)
187     {
188         int k, j;
189         double max = fabs(a[i][i]);
190         int tmp = 0;
191 
192         for (k = i+1; k < N; k++){
193             if(fabs(a[k][i]) > max){
194                 max = fabs(a[k][i]);
195                 tmp = k;
196             }
197         }
198 
199         if (tmp == 0) return 0;
200 
201         double t;
202         for(j = 0; j < N; j++){
203             t = a[i][j];
204             a[i][j] = a[tmp][j];
```

```
205             a[tmp][j] = t;
206
207             if (flags & REV) {
208                 t = rev[i][j];
209                 rev[i][j] = rev[tmp][j];
210                 rev[tmp][j] = t;
211             }
212         }
213
214         t = b[i];
215         b[i] = b[tmp];
216         b[tmp] = t;
217
218         return 1;
219     }
220
221     double **copy_1(double **res, double **a, int n)
222     {
223         res = calloc(n, sizeof(double *));
224         for (int h = 0; h < n; h++)
225             res[h] = calloc(n, sizeof(double));
226
227         for (int i = 0; i < n; i++)
228             for (int j = 0; j < n; j++)
229                 res[i][j] = a[i][j];
230
231         return res;
232     }
233
234     double *copy_2(double *res, double *f, int n)
235     {
236         res = calloc(n, sizeof(double));
237         for (int i = 0; i < n; i++)
238             res[i] = f[i];
239         return res;
240     }
241
242     double **create_rev(double **rev, int n)
243     {
244         rev = calloc(n, sizeof(double));
245         for (int i = 0; i < n; i++)
246             rev[i] = calloc(n, sizeof(double));
247         for (int i =0; i < n; i++)
248             rev[i][i] = 1. ;
249
250         return rev;
251     }
252
253     double norm(double **a, int n)
254     {
255         double res = 0. ;
256         for(int i = 0; i < n; i++)
257             for(int j = 0; j < n; j++)
258                 res += a[i][j] * a[i][j];
259         res = sqrt(res);
260         return res;
261     }
262
263
264     void init_matr(FILE *fin, double **A, double *f, int n, int mode)
265     {
266         double q = 1.001 - 2 * M * 0.001;
267
268         for (int i = 0; i < n; i++)
269             A[i] = calloc(n, sizeof(double));
270
271         switch (mode) {
272             case FILES:
273
```

```
274             for (int i = 0; i < n; i++) {
275                 for (int j = 0; j < n; j++) {
276                     fscanf(fin, "%lf", &A[i][j]);
277                 }
278             }
279 
280 
281             for (int i = 0; i < n; i++)
282                 fscanf(fin, "%lf", &f[i]);
283 
284             break;
285 
286         case FORMULES:
287             for (int i = 0; i < n; i++) {
288                 for (int j = 0; j < n; j++) {
289                     if (i == j) {
290                         A[i][j] = pow(q-1, (double)(i+j));
291                     } else {
292                         A[i][j] = pow(q, (double)(i+j)) + 0.1 * (j - i);
293                     }
294                 }
295             }
296 
297             double x = pi/2; // случайный параметр для генерация вектора f
298             for (int i = 0; i < n; i++)
299                 f[i] = fabs(x - N/10.) * i * sin(x);
300 
301             break;
302 
303         default:
304             return;
305     }
306 
307     return;
308 }
309 
310 
311 void check_print(double **A, double *f, int n)
312 {
313     printf("A:\n");
314     for (int i = 0; i < n; i++) {
315         for (int j = 0; j < n; j++) {
316             printf("%lf ", A[i][j]);
317         }
318         printf("\n");
319     }
320     printf("\nf: ");
321     for (int i = 0; i < n; i++)
322         printf("%lf ", f[i]);
323     printf("\n");
324 }
325 
```