```
1       /*
          File. opredeli.c

            opredeli.c

          Задача.

            Найти определитель квадратной матрицы N-го порядка методом
            исключения Гаусса.

            Задача для решения предложена Маратом 25.10.09

            Теория
            ------
            Определитель (детерминант) имеют только квадратные матрицы.

            Определитель треугольной (или диагональной) матрицы -
            это произведение ее диагональных элементов.

            Алгоритм. Метод исключения Гаусса.
            ----------------------------------
            Числа первой строки добавляются или вычитаются из оставшихся
            строк до получения нулей в каждом первом столбце, кроме первой
            строки. Затем числа из второй строки добавляются или
            вычитаются из оставшихся нижних строк до получения нулей в
            каждом втором столбце под диагональю. По окончании этой
            процедуры вы получите верхнюю треугольную матрицу.

            Конкретный элемент на диагонали, с которым вы работаете, чтобы
            устранить поддиагональные элементы, называется ведущим
            элементом.

            Определитель треугольной матрицы - это произведение ее
            диагональных элементов.

            Определение.
            ------------
            Назовем элементарными преобразованиями матриц следующие
            действия над ними:
             - перестановка строк или столбцов;
             - умножение строки или столбца на число, отличное от нуля;
             - добавление к одной из строк другой строки, умноженной на
               число
               или
               добавление добавление к одному из столбцов другого столбца,
               умноженного на число.

            Теорема 1.
            ----------
            При элементарных преобразованиях определитель матрицы
```

не меняется.

Теорема 2. [очень важная, про нее все забывают :-(]

Если поменять местами два столбца (строки), то определитель
изменит знак. [MSP, 30.10.09, 18:12]

Следствие

Если все элементы какого-нибудь столбца (строки) равны нулю,
то сам определитель равен нулю.

АЛГОРИТМ НАХОЖДЕНИЯ ОПРЕДЕЛИТЕЛЯ МАТРИЦЫ
---

1. если матрица A нулевая,
   то
      d = A [1][1]
   иначе
     с помощью перестановки строк и столбцов матрицы
     добиваемся того, чтобы в левом верхнем углу матрицы
     стоял ненулевой элемент.

2. Первую строку оставляем без изменений. Ко второй строке
   прибавляем первую, умноженную на число
     -A [2, 1] / A [1, 1].
   В результате получим, что первый элемент второй строки
   равен 0.
   Затем к третьей строке прибавляем первую строку, умноженную
   на число -A [3, 1] / A [1, 1].
   Этот процесс продолжаем до тех пор, пока не получим нуль
   в первом столбце последней строки.

   Первую и вторую строки оставляем без изменений. К третьей
   строке прибавляем вторую, умноженную на число:
   -A [3, 2] / A [2, 2].
   В результате получим, что второй элемент третьей строки
   равен 0.
   Затем к четвертой строке прибавляем вторую строку,
   умноженную на число -A [4, 2] / A [2, 2] и т.д.
   Этот процесс продолжаем до тех пор, пока не получим нуль
   уже во втором столбце последней строки.

   Перестановкой строк и столбцов с номерами,
   большими двух, добиваемся, чтобы третий элемент
   третьей строки был отличен от нуля.

   Далее, добавлением третьей строки, умноженной на
   соответствующее число, к строкам с большими номерами
   получаем нули в третьем столбце, начиная с

четвертого элемента и так далее по циклу.

На каком-то этапе мы придем к матрице треугольного вида, у которой ненулевые элементами на главной диагонали.

Представленный ниже алгоритм (код) мой.

Тесты.

1. 0 0 0 0
   0 0 0 0
   0 0 0 0
   0 0 0 0

   d = 0

2. 1 3
   4 5

   d = -7

   1 -3
   18 1

   d = 55

3. 2 1 3
   5 3 2
   1 4 3

   d = 40

   -3 0 1
   -5 2 4
   0 3 7

   d = -21

   1 2 3
   4 5 6
   7 8 9

   d = 0

4. 12314 16536 20537
   6157 8268 10268
   513   689   126

   d = 689

5. 1 1 1 0
   1 1 0 1
   1 0 1 1
   0 1 1 1

   d = -3

   0 0 0 1
   0 0 2 3
   0 3 1 4
   4 1 2 5

   d = 24

6. 2 3 4 2
   1 2 2 1
   2 3 4 1
   1 4 3 2

   d = 1

7. 35436 46343 22429
   17718 23171 11214
   5906 7723 3737

```
   d = 5906

8.    0    1    1
      0    2    3
   5906 7723 3737

   d = 5906

9. 1 0 0 0 0 0
   1 2 0 0 0 0
   1 2 3 0 0 0
   1 2 3 4 0 0
   1 2 3 4 5 0
   1 2 3 4 5 6

   d = 720

10. 1 1 1 1 1        2  0 -1 3 4
    2 2 2 2 2        0  1  0 2 1
    3 3 0 0 0       -1  0  1 5 3
    4 4 0 0 0       -3  1 -2 0 2
    5 5 0 0 0       -1 -1  2 5 1

    d = 0           d = -66

11. 1  2  3  4  5
    2  3  7 10 13
    3  5 11 16 21
    2 -7  7  7  2
    1  4  5  3 10

    d = 52

12. 1  0 -1 2  0  0
    0  0 -1 0 -2  1
    2 -1  0 1  3  1
    0  3  2 0 -1 -1
    1  0  2 2 -1  0
    1  2  0 0 -2  0

    d = 89

13. 2 -5 3  1
    3 -7 3 -1
    5 -9 6  2
    4 -6 3  1

    d = 18

14. (последний тест)
```

```
            1 -1 2 -2 1
            0  0 0  1 0
           -1  0 3 -1 3
            1 -1 3  0 1
            2 -2 7 -2 1

            d = 1

        15. 1 2 3 4
            0 2 3 4
            0 0 4 5
            0 0 0 6

            d = 48

      Решение.
        Сергей Митрофанов,
        Центр новых информационных технологий "Северная Звезда"
        гимназии "Лаборатория Салахова",
        28.10.09, 12:06;
        29.10.09, 10:00;
        30.10.09, 10:22-19:15.
    */

172 # include <stdio.h>
173 # include <conio.h>
174 # include <stdlib.h>
175 # include <time.h>
176 # include <math.h>

177 // прототипы
178 void random_matrix (double *, int);
179 void print_m (double *, int);
180 int f_treug (double *, int);
181 int f_null_str (double *, int);
182 int f_null_stb (double *, int);
183 double det (double *, int);
184 double treug (double *, int);
185 void perest_stb (double *, int, int);
186 void make_null_stb (double *, int, int);
187 //void make_null (double *, int);

188 // 1.
189 void random_matrix (
190                       double * A, // адрес первого элемента матрицы
191                       int N // порядок матрицы
192                     )
193 {
194   // MSP, 29.10.09, 09:34
195   // Зададим значения элементов матрицы случайным образом
```

```
196         int i, j; // индексы матрицы

197         for (i = 0; i < N; i ++)
198           for (j = 0; j < N; j ++)
199             * (A + i * N + j) = random (10) - 1;

200       } // random_matrix (double *, int)


201       // 2.
202       void print_m (
203                     double * A, // адрес первого элемента матрицы
204                     int N // порядок матрицы
205                   )
206       {
207         // MSP, 29.10.09, 09:40
208         // печать матрицы на экран дисплея

209         int i, j; // индексы матрицы

210         //printf ("\nМатрица:\n");
211         for (i = 0; i < N; ++ i)
212         {
213           for (j = 0; j < N; ++ j)
214             printf ("%9.2lf", * (A + i * N + j));
215           printf ("\n");
216         }
217       }

218       // 3.
219       int f_treug (
220                     double * A, // адрес первого элемента матрицы
221                     int N // порядок матрицы
222                   )
223       {
224         // MSP, 29.10.09, 09:55
225         // Проверка матрицы на "треугольность"
226         // 1 - матрица треугольная (диагональная)

227         int i, j; // индексы матрицы

228         // просмотрим все диагональные элементы, кроме нижнего правого
229         // просмотрим столбец сверху-вниз
230         // последний столбец просматривать не надо
231         for (j = 0; j < N - 1; j ++)
232           // под диагональным элементом должны быть все нули
233           for (i = j + 1; i < N; ++ i)
234             if (* (A + i * N + j) != 0)
235               return 0; // матрица не является диагональной
```

```
236         return 1; // матрица треугольная
237       } // int f_treug (double *, int)


238       // 4.
239       int f_null_str (
240                       // адрес первого элемента матрицы
241                       double * A,
242                       int N // порядок матрицы
243                     )
244       {
245         // MSP, 29.10.09, 12:18
246         // есть ли в матрице хотя бы одна нулевая строка?
247         // 1 - обнаружена нулевая строка

248         int
249           S, // счетчик нулей
250           i, j; // индексы матрицы

251         //printf ("null_str");

252         // ввод матрицы
253         for (i = 0; i < N; ++ i)
254         {
255           S = 0;
256           for (j = 0; j < N; ++ j)
257             if (* (A + i * N + j) == 0.0)
258               ++ S;

259           if (S == N)
260             return 1;
261         }

262         return 0;
263       } // int f_null_str (double *, int)


264       // 5.
265       int f_null_stb (
266                       // адрес первого элемента матрицы
267                       double * A,
268                       int N // порядок матрицы
269                     )
270       {
271         // MSP, 29.10.09, 12:18
272         // есть ли в матрице нулевой столбец?
273         // 1 - обнаружена нулевой столбец

274         int
275           S, // счетчик нулей
276           i, j; // индексы матрицы
```

```
277         // ввод матрицы
278         for (j = 0; j < N; ++ j)
279         {
280           S = 0;
281           for (i = 0; i < N; ++ i)
282             if (* (A + i * N + j) == 0.0)
283               ++ S;

284           if (S == N)
285             return 1;
286         }

287         return 0;
288       } // int f_null_stb (double *, int)


289       // 7.
290       double treug (
291                     double * A, // адрес первого элемента матрицы
292                     int N // порядок матрицы
293                   )
294       {
295         // MSP, 29.10.09, 14:20
296         // приведение матрицы к треугольному виду

297         int
298           znak = 1, // знак определителя
299           i, j, // индексы матрицы
300           stb; // номер столбца матрицы

301         double d; // значение определителя

302         stb = 0;
303         while (stb < N - 1)
304         {
305           // В матрице нет нулевых столбцов и строк.
306           //
307           // В первой строке есть хотя бы один ненулевой элемент,
308           // сделаем его первым элементом первой строки,
309           // переставляя столбцы.
310           //
311           // Добьемся того, чтобы в зависимости от номера обнуляемого
312           // внизу столбца, на месте A [k, k] стоял ненулевой элемент.
313           //
314           // Перестановкой строк и столбцов матрицы с номерами, большими
315           // k, добиваемся, чтобы k-тый элемент k-той строки был отличен
316           // от нуля: A [k, k] <> 0.
317           if (* (A + stb * N + stb) == 0.0)
318           {
319             perest_stb (A, stb, N);
```

```
320             znak *= -1; // определитель поменяет знак!
321             //printf
322               //("Поставили ненулевой элемент в A [%i][%i]", stb, stb);
323             //print_m (A, N);
324             //getch ();
325           }

326           // Первую строку оставляем без изменений.
327           // Ко второй строке прибавляем первую, умноженную на число
328           // -A [2, 1] / A [1, 1].
329           // Затем к третьей строке прибавляем первую строку, умноженную
330           // на число -A [3, 1] / A [1, 1].
331           // Этот процесс продолжаем до тех пор, пока не получим нуль
332           // в первом столбце последней строки.
333           if (stb < N - 1)
334           {
335             //printf ("Обнулим низ %i столбца", stb);
336             make_null_stb (A, stb, N);
337           }

338           //printf ("iz treug (): ");
339           //print_m (A, N);

340           // все машинные нули превратим в настоящий нуль
341           //make_null (A, N);

342           // если в матрице найдется нулевая строка
343           if (f_null_str (A, N) == 1)
344           {
345             //printf ("\nB матрице обнаружена нулевая строка,\n");

346             return 0.0;
347           }

348           // если в матрице найдется нулевой столбец
349           if (f_null_stb (A, N) == 1)
350           {
351             //printf ("\nB матрице обнаружен нулевой стодбец,\n");

352             return 0.0;
353           }

354           ++ stb;
355         }

356         // матрица приведена к треугольному виду,
357         // найдем ее определитель,
358         // найдем произведение всех элементов на главной диагонали
359         d = 1.0;
360         for (i = 0; i < N; ++ i)
361           d = d * (* (A + i * N + i));
```

```
362         // MSP, 30.10.09, 18:17
363         d = znak * d;

364         //printf ("**** d = %10.5lf\n", d);

365         return d;
366       } // double treug (double *, int)


367       // 8.
368       void perest_stb (
369                          double * A, // адрес матрицы
370                          int stb, // номер переставляемого столбца
371                          int N // размер матрицы
372                        )
373       {
374         // MSP, 29.10.09, 16:28
375         // Переставим столбцы так, чтобы в верхнем углу матрицы стоял
376         // ненулевой элемент

377         int
378           i, j, // индексы матрицы
379           j_stb; // номер столбика в котором нет нуля

380         double t; // для временного хранения элемента матрицы

381         //j_stb = 0;
382         for (j = stb; j < N; ++ j)
383           if (* (A + stb * N + j) != 0.0)
384           {
385             j_stb = j;

386             break;
387           }

388         // переставляем столбец j_stb на новое место
389         for (i = 0; i < N; ++ i)
390         {
391           t = * (A + i * N + j_stb);
392           * (A + i * N + j_stb) = * (A + i * N + stb);
393           * (A + i * N + stb) = t;
394         }

395         //printf ("\nпосле перестановки столбика: ");
396         //print_m (A, N);
397         //getch ();
398       } // void perest_stb (int, int)


399       // 9.
```

```
400     void make_null_stb (
401                         double * A, // адрес матрицы
402                         int stb, // номер обнуляемого столбца
403                         int N // размер матрицы
404                        )
405     {
406       // MSP, 30.10.09, 11:46
407       // обнулим низ столбца stb

408       // Алгоритм обнуления 1-го столбца
409       // -------------------------------
410       //
411       // Первую строку оставляем без изменений. Ко второй строке
412       // прибавляем первую, умноженную на число -A [2, 1] / A [1, 1].
413       // Затем к третьей строке прибавляем первую строку,
414       // умноженную на число -A [3, 1] / A [1, 1].
415       // Этот процесс продолжаем до тех пор, пока не получим нуль
416       // в первом столбце последней строки

417       int i, j; // индексы матрицы

418       double koef; // число, на которое умножаем элементы строки

419       for (i = stb + 1; i < N; ++ i)
420       {

421         koef = -1 * (* (A + i * N + stb)) / (* (A + stb * N + stb));

422         //printf ("koef = %10.5lf\n", koef);

423         /*
          printf
            ("-1 * (* A + %i * %i + %i)"
             " / (* (A + %i * %i + %i) = %5.1f\n",
             i, N, stb, stb, N, stb, koef);
          getch ();
          */

430         for (j = 0; j < N; ++ j)
431         {
432           //printf ("stb_j = %i  %10.5lf\n", j,
433           //       (* (A + stb * N + j)) * koef);
434           //getch ();

435           * (A + i * N + j) =
436             * (A + i * N + j) + ((* (A + stb * N + j)) * koef);
437         }
438         //printf ("\nпереставили %i строчку: ", i);
439         //print_m (A, N);
440         //getch ();
441       }
```

```
442       } // void make_null_stb (int)


443       /*
          // 10.
          void make_null (
                          double * A, // адрес матрицы
                          int N // размер матрицы
                         )
          {
            // MSP, 30.10.09, 12:01
            // Если, вдруг, какой-то элемент матрицы стал настолько мал, что
            // ничем не отличается от нуля, то сделаем его настоящим нулем

            int i, j; // индексы матрицы

            // точность вычислений, для поиска нуля
            double eps = pow (10.0, -10.0);


            for (i = 0; i < N; ++ i)
              for (j = 0; j < N; ++ j)
                if (fabs (* (A + i * N + j)) < eps)
                  * (A + i * N + j) = 0.0;

          } // make_null ()
          */

462       // 6.
463       double det (
464                   double * A, // адрес первого элемента матрицы
465                   int N // порядок матрицы
466                  )
467       {
468         // MSP, 29.10.09, 10:54
469         // вычисление детерминанта треугольной матрицы

470         int i, j; // индексы матрицы

471         double
472           C [3][3], // матрица третьего порядка
473           B [2][2];

474         double d = 0.0; // значение определителя-детермината

475         if (N == 1)
476           return * A;

477         // если матрица уже треугольная, то найдем ее определитель
478         if (f_treug (A, N) == 1)
```

```
479         {
480           //printf ("\nМатрица треугольная,\n");

481           d = 1.0;
482           for (i = 0; i < N; i ++)
483             d = d * (* (A + i * N + i));

484           return d;
485         }

486         // если в матрице найдется нулевая строка
487         if (f_null_str (A, N) == 1)
488         {
489           //printf ("\nВ матрице обнаружена нулевая строка,\n");
490           d = 0.0;

491           return d;
492         }

493         // если в матрице найдется нулевой столбец
494         if (f_null_stb (A, N) == 1)
495         {
496           //printf ("\nВ матрице обнаружен нулевой стодбец,\n");
497           d = 0.0;

498           return d;
499         }

500         if (N == 2)
501         {
502           for (i = 0; i < N; ++ i)
503             for (j = 0; j < N; ++ j)
504               B [i][j] = * (A + i * N + j);

505           return B [0][0] * B [1][1] - B [0][1] * B [1][0];
506         }

507         // если матрица третьего порядка
508         if (N == 3)
509         {
510           for (i = 0; i < N; ++ i)
511             for (j = 0; j < N; ++ j)
512               C [i][j] = * (A + i * N + j);

513           return
514               C [0][0] * C [1][1] * C [2][2]
515             + C [0][1] * C [1][2] * C [2][0]
516             + C [1][0] * C [2][1] * C [0][2]
517             - C [0][2] * C [1][1] * C [2][0]
518             - C [0][1] * C [1][0] * C [2][2]
519             - C [0][0] * C [1][2] * C [2][1];
```

```
520         }

521         // приведем матрицу к треугольному виду
522         if (N > 3)
523           return treug (A, N);

524         printf ("\nНе умею находить определитель такой матрицы :-(\n");

525         return d;
526       } // det (double *, int)


527       int main ()
528       {
529         int
530           i, j, // индексы матрицы
531           N; // порядок данной матрицы

532         double * A; // адрес первого элемента матрицы

533         double
534           d, // значение детерминанта матрицы
535           t; // элемент матрицы

536         clrscr ();

537         randomize ();

538         printf ("Порядок матрицы: ");
539         scanf ("%i", & N);
540         fflush (stdin);

541         if (N <= 0)
542           return -1;

543         A = (double *) malloc (N * N * sizeof (double));

544         // ввод матрицы по датчику случайных чисел
545         //random_matrix (A, N);

546         // ввод матрицы
547         for (i = 0; i < N; ++ i)
548           for (j = 0; j < N; ++ j)
549           {
550             printf ("e [%i][%i] = ", i, j);
551             scanf ("%lf", & t);
552             fflush (stdin);
553             * (A + i * N + j) = t;
554           }

555         printf ("\nДана матрица:\n");
```

```
556         print_m (A, N);

557         d = det (A, N);

558         printf ("\nДля подсчета определителя,\n");
559         printf ("матрицу преобразовали:\n");
560         print_m (A, N);

561         printf ("\nОпределитель равен %10.5lf\n", d);

562         // MSP, 01.11.09, 12:36
563         free (A);

564         getch ();

565         return 0;
566       }
```

Listing данной задачи опубликован в сети Internet по адресу
http://www.Best-Listing.ru/color-2-task-109.html

Sergey Mitrofanov, 31.10.13, 21:50
E-mail: infostar @ mail.ru