

1 Program rang_m;

2 {

Задача. Рассчитать ранг введенной матрицы.
Автоматизировать ввод матрицы.
Создать возможность распечатки ранга текущей матрицы
или всех расчетов матриц, производимых в данной
программе.

Решение. MSP for Ярослав Казанжи,
CNIT "North Star",
10.05.05, 13:00
11.05.05, 10:03-18:00

}

13 {

Теория.

Определение 1.

Пусть дана матрица A размером $M \times N$ и число K , не
превосходящее наименьшего из чисел M и N :

$$K \leq \min(M, N).$$

Выберем произвольно K строк матрицы A и K столбцов (номера
строк могут отличаться от номеров столбцов).

Определитель матрицы, составленный из элементов, стоящих на
пересечении выбранных K строк и K столбцов, называется
МИНОРОМ ПОРЯДКА K матрицы A .

Определение 2.

РАНГОМ МАТРИЦЫ A называется наибольший из порядков миноров
матрицы A , отличный от нуля. Ранг нулевой матрицы считается
равным нулю.

Теорема 1.

Ранг матрицы равен максимальному числу ее столбцов (или
строк), образующих линейно-независимую систему.

Определение 3.

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 к одному из столбцов другого
столбца, умноженного на число.

Теорема 2.

При элементарных преобразованиях ранг матрицы не меняется.

Алгоритм вычисления ранга матрицы похож на алгоритм вычисления

определителя и заключается в том, что с помощью элементарных преобразований матрица приводится к простому виду, для которого найти ранг не представляет труда. Так как при каждом преобразовании ранг не меняется, то, вычислив ранг преобразованной матрицы, мы, тем самым, находим ранг исходной матрицы.

АЛГОРИТМ НАХОЖДЕНИЯ РАНГА МАТРИЦЫ

1. если
 матрица A нулевая, то $\text{rang}_A = 0$
 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 на первом месте в последней строке.
3. Если все строки, начиная со второй, в полученной матрице нулевые, то ее ранг равен 1, так как есть минор первого порядка, отличный от нуля $A [1, 1]$.

В противном случае перестановкой строк и столбцов матрицы с номерами, большими ЕДИНИЦЫ, добиваемся, чтобы второй элемент второй строки был отличен от нуля: $A [2, 2] \neq 0$.

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 на втором месте в последней строке.

4. Если все строки, начиная с третьей, нулевые, то $\text{rang}_A = 2$, так как минор второго порядка не равен нулю.

В противном случае 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 т.д. по циклу.

5. Все нулевые строки и столцы уничтожаем в цикле!
6. На каком-то этапе получим матрицу, у которой минор первых k строк и первых k столбцов является определителем треугольной матрицы с ненулевыми элементами на диагонали. Ранг такой матрицы равен k .

```
rang_A = k.
```

```
}
```

```
95  Uses Crt;

96  Const
97      Exp = 0.0000000001; { точность вычислений, для поиска нуля }
98      N = 5; { число строк данной матрицы }
99      M = 5; { число столбцов данной матрицы }

100  Type
101      bits = 0..1;
102      matrix = array [1..N, 1..M] of real;

103  { для тестов }
104  Const
105      {
106      A
107      : matrix = ((0, 0, 0, 0), (0, 0, 0, 0), (0, 0, 0, 0));
108      }

109      {
110      A
111      : matrix = ((1, 2, -1, 0), (3, 4, -5, 6), (5, -2, -3, -4));
112      }

113      {
114      A
115      : matrix = ((0, 2, -1, 0), (0, 0, 0, 0), (0, -2, -3, -4));
116      }

117      {
118      A
119      : matrix = ((5, 0, 3, 0), (1, 0, 0, 1), (0, 0, 0, 0));
120      }

121  { великолепный пример!}
```

```

122     A
123     : matrix = (
124         ( 1, -1, 2, 3, 4),
125         ( 1, -1, -1, 2, 0),
126         ( 1, -1, 2, 1, 3),
127         ( 1, -1, -1, 3, 4),
128         ( 3, -7, 8, 9, 13)
129     );
130     { rang = 5 }

131     {
132     A
133     : matrix = (
134         ( 1, -1, 2, 3, 4),
135         ( 2, 1, -1, 2, 0),
136         ( -1, 2, 1, 1, 3),
137         ( 1, 5, -8, -5, -12),
138         ( 3, -7, 8, 9, 13)
139     );
140     }

141     {
142     A
143     : matrix = (
144         (1, -1, 2, 3, 4),
145         (1, -1, 2, 3, 4),
146         (1, -1, 2, 3, 4),
147         (1, -1, 2, 3, 4),
148         (1, -1, 2, 3, 4)
149     );
150     }

151     {
152     A
153     : matrix = (
154         (2, 2, -1, 0),
155         (3, 1, 4, -1),
156         (5, 9, -13.5, 1),
157         (3, 5, -7, 1)
158     );
159     }

160     {
161     A
162     : matrix = (
163         (1, 3, -4, 5),
164         (2, 5, 2, 1),
165         (2, 4, 4, -3),
166         (0, 2, -1, -4)
167     );
168     }

169     {

```

```

A
: matrix = (
          (3, 2, 1, -3, 1),
          (2, -1, -1, 0, 3),
          (4, 5, 2, -6, -5),
          (1, -4, -2, 3, 7)
        );
} { rang = 4 }

178
{
A
: matrix = (
          (3, 2, 1, -3, -1),
          (2, -1, -1, 0, 3),
          (4, 5, 2, -6, -5),
          (1, -4, -2, 3, 7)
        );
} { rang = 3 }

187
{
A
: matrix = (
          (3, -4, 5),
          (2, -3, -6),
          (1, -1, 3)
        );
} { rang = 3}

195
{
A
: matrix = (
          (1, 4, 1, -2),
          (2, 1, -1, 3),
          (1, -3, 2, -1),
          (1, -3, 10, -13)
        );
} { rang = 3 }

204
{
A
: matrix = (
          (2, 4, 3, -3, 5),
          (1, -2, 1, 5, 3),
          (1, -2, 4, -34, 0)
        );
} { rang = 3 }

212
{
A
: matrix = (
          (2, 3, 1, -1),
          (3, 1, 4, 2),
          (1, 2, 3, -1),
          (1, -4, -7, 5)

```

```

                );
221   } { rang = 3 }
      {
      A
      : matrix = (
                (2, 1, 1, 0, -2),
                (-1, 0, 3, -5, 1),
                (1, 1, -3, 4, -2),
                (2, 2, -1, 3, -1)
                );
      } { rang = 4}

230   {
      A
      : matrix = (
                (1, 3, -1, 2, 4),
                (0, -1, 2, 3, 1),
                (1, 2, -3, -1, -3),
                (1, 4, 1, 5, 11),
                (-1, -4, 7, 5, 15)
                );
      } { rang = 4 }

240   Var
241     rang_A, { ранг матрицы }
242     n_str, { число строк текущей матрицы }
243     n_stb, { число столбцов текущей матрицы }
244     i, { индекс строки }
245     j, { индекс столбца }
246     k { номер неполного нулевого столбца }
247     : word;

248     {
249     A { данная матрица }
250     {
251     : matrix;
252     }

253   Procedure print_m;
254   {
255     печать матрицы на экран дисплея
256   }

257   var
258     i, j { индексы матрицы }
259     : integer;

260   begin
261     WriteLn;
262     for i := 1 to n_str do

```

```

263         begin
264             for j := 1 to n_stb do
265                 Write (A [i, j]:7:2, ' ');
266                 WriteLn;
267             end;
268         WriteLn;
269         Write ('Нажмите на Enter... ');
270         ReadLn;
271         WriteLn;
272     end;

273     Function null_matrix
274         : bits;
275     {
        Проверка матрицы на нуль.
        Если матрица нулевая, то null_matrix = true
    }

279     var
280         i, j { индексы матрицы }
281         : word;

282     begin
283         null_matrix := 0;
284         for i := 1 to n_str do
285             for j := 1 to n_stb do
286                 if A [i, j] <> 0
287                     then
288                         begin
289                             null_matrix := 1;
290
291                             Break;
292                         end;
293             end;

294     Procedure print_rang_m;
295     {
        печать ранга матрицы
    }

297     begin
298         WriteLn;
299         WriteLn ('Ранг матрицы равен: ', rang_A);
300         WriteLn;
301     end;

302     Procedure prov_null_matrix;
303     {

```

```

    проверка на появление нуль-матрицы
}

306 begin
307   { если исходная матрица - нулевая }
308   {
        такое, например, может случиться тогда, когда останется
        всего одна строка и все ее элементы равны нулю
    }
312   if null_matrix = 0
313     then
314       begin
315         rang_A := 0;
316         print_rang_m;
317         ReadLn;
318         Halt (0);
319       end;
320   end;

321 Function null_str (
322     str { номер строки }
323     : word
324     )
325     : bits;
326 {
    проверка на нуль-строку.
    Если строка состоит только из нулей, то null_str = 0
}

330 var
331   j { номер столбца }
332   : word;

333 begin
334   null_str := 0;
335   for j := 1 to n_stb do
336     if A [str, j] <> 0
337       then
338         begin
339           null_str := 1;
340           Break;
341         end;
342   end;

343 Function null_stb (
344     stb { номер столбца }
345     : word
346     )
347     : bits;

```



```

348     {
        проверка на нуль-столбец.
        Если строка состоит только из нулей, то null_stb = 0
    }

352     var
353         i { номер строки }
354         : word;

355     begin
356         null_stb := 0;
357         for i := 1 to n_str do
358             if A [i, stb] <> 0
359                 then
360                     begin
361                         null_stb := 1;
362                         Break;
363                     end;
364     end;

365     Procedure del_str (
366         str { номер удаляемой строки }
367         : word
368         );
369     {
        удалить строку
    }

372     var
373         i, j { индексы матрицы}
374         : word;

375     begin
376         for i := str + 1 to n_str do
377             for j := 1 to n_stb do
378                 A [i - 1, j] := A [i, j];

379         { изменим глобальную переменную - число строк в данном массиве }
380         Dec (n_str);
381     end;

382     Procedure del_stb (
383         stb { номер удаляемого столбца }
384         : word
385         );
386     {
        удалить столбец с номером stb
    }

```

```

389     var
390         i, j { индексы матрицы}
391         : word;

392     begin
393         for i := 1 to n_str do
394             for j := stb + 1 to n_stb do
395                 A [i, j - 1] := A [i, j];

396         { изменим глобальную переменную - число строк в данном массиве }
397         Dec (n_stb);
398     end;

399     Procedure perest_stb (
400         stb { номер обнуляемого столбца }
401         : word
402         );
403     {
404         Переставим столбцы так, чтобы в верхнем углу матрицы стоял
405         ненулевой элемент
406     }

407     var
408         i, j, { индексы матрицы }
409         j_stb { номер столбика, который нужно переставить }
410         : word;

411         temp { для временного хранения элемента матрицы}
412         : real;

413     begin
414         j_stb := 0;
415         for j := stb to n_stb do
416             if A [stb, j] <> 0
417                 then
418                     begin
419                         j_stb := j;
420                         Break;
421                     end;

422         {
423             переставляем столбец весь на новое место
424         }
425         {
426             WriteLn ('j_stb = ', j_stb);
427         }
428         for i := 1 to n_str do
429             begin
430                 temp := A [i, j_stb];
431                 A [i, j_stb] := A [i, stb];

```

```

432         A [i, stb] := temp;
433     end;

434     {
    for j := 1 to n_stb do
        if A [1, j] <> 0
            then
                begin
                    j_stb := j;
                    Break;
                end;

    for i := 1 to n_str do
        begin
            temp := A [i, j_stb];
            A [i, j_stb] := A [i, 1];
            A [i, 1] := temp;
        end;
    }
449 end;

450 Procedure make_null_stb (
451     stb { номер обнуляемого столбца }
452     : word
453 );

454 {
    Первую строку оставляем без изменений. Ко второй строке
    прибавляем первую, умноженную на число  $-A [2, 1] / A [1, 1]$ .
    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 на
    первом месте в последней строке.
}

462 var
463     i, j { индексы матрицы }
464     : word;

465     koef { число, на которое умножаем элементы строки }
466     : real;

467 begin
468     {
    for i := 2 to n_str do
        begin
            koef :=  $-A [i, 1] / A [1, 1]$ ;
            for j := 1 to n_stb do
                A [i, j] := A [i, j] + A [1, j] * koef;
            end;
        end;
    }

```

```

    }
476   for i := stb + 1 to n_str do
477     begin
478       koef := -A [i, stb] / A [stb, stb];
479       for j := 1 to n_stb do
480         A [i, j] := A [i, j] + A [stb, j] * koef;
481       end;
482   end;

483   Procedure delete_all_null_str;
484   {
      найдем и удалим все нулевые строки
   }

487   var
488     i { номер строки }
489     : word;

490   begin
491     i := 1;
492     while i <= n_str do
493       begin
494         if null_str (i) = 0
495           then
496             begin
497               WriteLn (i, ' строка - нулевая, удалим ее,');
498               del_str (i); { n_str - 1 }
499               {
                    проверим, вдруг, матрица нулевая
                }
502               prov_null_matrix;
503               {
                    печать полученной матрицы
                }
506               WriteLn ('получим матрицу:');
507               print_m;
508               i := 1;
509               Continue;
510             end;
511           {
                    если строчка не была нулевой, то рассмотрим следующую
                }
514             Inc (i);
515           end;

516   end;

517   Procedure delete_all_null_stb;
518   {

```

```

    найдем и удалим все нулевые столбцы
}

521  var
522      j { номер столбца }
523      : word;

524  begin
525      j := 1;
526      while j <= n_stb do
527          begin
528              if null_stb (j) = 0
529                  then
530                      begin
531                          WriteLn (j, ' столбец - нулевой, удалим его,');
532                          del_stb (j); { n_stb - 1 }
533                          {
534                              проверим, вдруг, матрица нулевая
535                          }
536                          prov_null_matrix;
537                          {
538                              печать полученной матрицы
539                          }
540                          WriteLn ('получим матрицу:');
541                          print_m;
542                          j := 1;
543                          Continue;
544                      end;
545                      {
546                          если столбик не был нулевым, то рассмотрим следующий
547                      }
548                      Inc (j);
549                  end;
550      end;

551  Procedure make_null;
552  {
553      Если, вдруг, какой-то элемент матрицы стал настолько мал, что
554      ничем не отличается от нуля, то сделаем его настоящим нулем
555  }

556  var
557      i, j { индексы матрицы }
558      : word;

559  begin
560      for i := 1 to n_str do
561          for j := 1 to n_stb do
562              if Abs (A [i, j]) < Exp
563                  then

```

```

564             A [i, j] := 0;
565     end;

566     Procedure random_matrix;
567     {
        зададим значения элементов матрицы случайным образом
    }

570     var
571     i, j { индексы матрицы }
572     : word;

573     begin
574     for i := 1 to n_str do
575     for j := 1 to n_stb do
576     if Random (2) = 0
577     then
578     A [i, j] := -Random (5)
579     else
580     A [i, j] := Random (5);
581     end;

582     Begin
583     ClrScr;
584     Randomize;

585     n_str := N;
586     n_stb := M;

587     {
        Write ('Сколько строк у матрицы: ');
        ReadLn (n_str);

        Write ('Сколько столбцов у матрицы: ');
        ReadLn (n_stb);

        {
            автоматизируем ввод матрицы A
        }
595     {
        random_matrix;
    }

598     WriteLn ('Дана матрица:');
599     print_m;

600     {
        а если матрица нулевая?

```

```

    то ее ранг равен 0
}
604 prov_null_matrix;

605 {
    найдем и удалим все нулевые строки
}
608 delete_all_null_str;

609 {
    найдем и удалим все нулевые столбцы
}
612 delete_all_null_stb;

613 {
    если матрица превратилась в A [1..1, 1..1],
    то ее rang_A = 1
}
617 if (n_str = 1)
618     and
619     (n_stb = 1)
620     then
621     begin
622         rang_A := 1;
623         print_rang_m;
624         ReadLn;
625         Exit;
626     end;

627 {
    Первую строку оставляем без изменений. Ко второй строке
    прибавляем первую, умноженную на число  $-A [2, 1] / A [1, 1]$ .
    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 на
    первом месте в последней строке.
}
635 k := 1;
636 while k <= n_str do
637 {
    while k <= n_str - 1 do
}
640     begin
641     {
        Теперь в матрице нет нулевых столбцов и строк.
        В первой строке есть точно хотя бы один ненулевой элемент,
        сделаем его первым элементом первой строки, переставляя
        столбцы.

        Добьемся того, чтобы в зависимости от номера обнуляемого
        внизу столбца, на месте A [k, k] стоял ненулевой элемент.

```

Перестановкой строк и столбцов матрицы с номерами, большими k , добиваемся, чтобы k -тый элемент k -той строки был отличен от нуля: $A[k, k] \neq 0$.

```
654 }
    {
    WriteLn ('A [k, k] = ', A [k, k]);
    }
657 if A [k, k] = 0
658     then
659         begin
660             perest_stb (k);
661             WriteLn
662                 ('Поставим ненулевой элемент в A [', k, ', ', k,
663                  ']');
664             print_m;
665         end;

666     if k < n_str
667         then
668             begin
669                 WriteLn ('Обнулим низ ', k, ' столбца: ');
670                 make_null_stb (k);
671             end;

672     print_m;
673     {
        все машинные нули превратим в настоящий нуль
    }
676     make_null;

677     delete_all_null_str;
678     delete_all_null_stb;
679     {
    WriteLn ('нулевой низ ', k, ' столбца сформирован...');
    ReadLn;

    WriteLn ('n_str = ', n_str, 'n_stb = ', n_stb);
    }
684     Inc (k);
685     {
    WriteLn ('k = ', k);
    }
688 end;

689 {
    Теперь в матрице нет ни одной нулевой строки и нулевого
    столбца.
```


Если все строки, начиная со второй, в полученной матрице нулевые, то ее ранг равен 1, так как есть минор первого порядка, отличный от нуля $A [1, 1]$

Получена матрица, у которой минор первых k строк и первых k столбцов является определителем треугольной матрицы с ненулевыми элементами на диагонали. Ранг такой матрицы равен k .

```
    rang_A = k.  
  }  
701  rang_A := n_str;  
702  print_rang_m;  
  
703  WriteLn ('Задача решена!');  
704  
705  ReadLn;  
706  End.
```

Listing данной задачи опубликован в сети Internet по адресу <http://www.Best-Listing.ru/color-1-task-594.html>

Sergey Mitrofanov, 28.08.14, 20:45

E-mail: infostar@mail.ru

© <http://www.Best-Listing.ru/>, 2006–2014