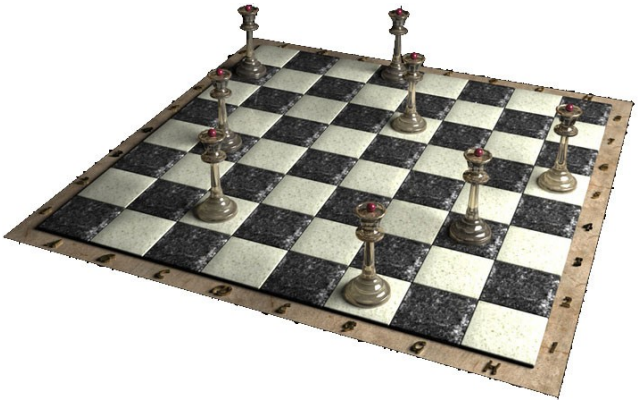
Как научить школьника писать рекурсивные процедуры?

Морозов В. В.

Умение создавать и использовать рекурсивные процедуры – неотъемлемая часть подготовки школьников к успешному выступлению на олимпиадах по программированию. Как научить школьника программировать с использованием рекурсий? Как научить его видеть ситуации, когда без рекурсий обойтись нельзя?



Одна из интереснейших задач программирования, которую следует решать с помощью рекурсий, как раз является **задача о ферзях**[1] — это классическая задача о расстановке ферзей на шахматной доске. Вот ее формулировка: расставить на обычной шахматной доске 8 ферзей так, чтобы ни один из них не бил другого.

Этой красивой, но сложной задачей можно зажечь интерес учеников, и, увлекая их за собой в решении этой задачи, показать, насколько эффективно можно использовать рекурсивные процедуры.

Постановка задачи. Написать компьютерную программу, которая находит:

а) хотя бы один из способов такой расстановки ферзей;

б) все способы расстановки ферзей на доске (поскольку способов расстановки может быть очень много, то желательно вывести все найденные решения в файл);

в) подсчитывает количество всевозможных решений.

Прежде чем начинать работу над составлением алгоритма решения задачи следует предложить ученику самому расставить ферзей на шахматной доске так, чтобы они не били друг друга, а также предложить решить ту же задачу, но на доске с меньшими размерами. Это позволит ученику увидеть, что не всегда удается довести процедуру расстановки ферзей до успешной расстановки, что приходится возвращаться на несколько ходов, и пробовать снова. Ученик почувствует необходимость использовать рекурсивную процедуру. Было бы здорово, если бы ученик самостоятельно сформулировал мысль, что без рекурсии при решении этой задачи на компьютере не обойтись.

После этого приступаем к обсуждению алгоритма.

[1] (Гик, 2009)

Искусство решать задачи – это искусство задавать себе вопросы. Почему так громко – «искусство»? Потому что, задавая себе легкие вопросы (мы ведь любим себя, не мучаем себя трудными неподъемными вопросами), шаг за шагом мы приходим к ответу на главный вопрос поставленной задачи, к решению проблемы. Итак, начиная обсуждать с учеником будущий алгоритм, склонившись над шахматной доской с ферзями в руках, начинаем задавать себе легкие вопросы.

Ставим первого ферзя в вертикаль A1-A8. Сколькими способами это можно сделать? Восемью способами. Каждый из способов расстановки первого ферзя – это одна из веток дерева решений. Пробуем поставить второго ферзя в вертикаль B1-B8. Сколькими способами это можно сделать? И снова каждый из способов – одна из веток дерева решений. Чем больше размер доски, тем более ветвистым оказывается дерево решений. Но не каждая ветка дерева решений заканчивается успешной расстановкой всех 8 ферзей.

Итак, будем писать рекурсивную процедуру, которая на вход получает доску с уже расставленными k ферзями в первых k вертикалях, возможно, что количество расставленных ферзей k=0, а на выходе эта процедура должна попытаться расставить очередного k+1-го ферзя в k+1-ю вертикаль всеми доступными способами. Причем после каждой удачной расстановки k+1-го ферзя в k+1-ю вертикаль процедура будет вызывать себя рекурсивно, и пытаться расставить следующего ферзя.

В случае, если k+1-й ферзь ставить уже некуда, то текущая ветвь решения закрывается. Если поставлен последний ферзь в последнюю вертикаль, то найдено одно из решений, и очередное решение запишем в файл.

Итак, начинаем реализовывать предлагаемый алгоритм. Каким образом будем организовывать хранение данных? Предлагаем ученикам для обсуждения два варианта.

- Можно объявить матрицу размером 8×8. Если поле чистое, то в соответствующей ячейке будем хранить 0, а если в поле поставили ферзя, то запишем в соответствующую ячейку, скажем, 8. Битые ферзями поля будем помечать единицами. Чистые небитые поля отмечаем нулями.
- Можно объявить массив из 8 элементов, пронумерованных литерами от ‘A’ до ‘H’, в которых хранится координаты выставленных ферзей.

Первый вариант позволяет использовать двумерную матрицу как наглядную модель шахматной доски. Второй вариант менее нагляден, несколько сложнее в реализации, но позволяет экономно расходовать память.

Будем расставлять ферзей, записывая в соответствующие ячейки квадратной матрицы число 8.

Таблица 1

`Program Queens;`
`uses crt;`
`const n=8;`
	Объявляем константу n – размер доски.

`Type` `  TBoard=array[1..n,1..n] of integer;`	Поскольку мы собираемся передавать рекурсивной процедуре матрицу – объявляем для этого тип TBoard.

`var p:TBoard;`
`  L:Boolean;`
`  i,k,d:integer;`
`  f:text;`
	Объявляем глобальные переменные:
	p – шахматная доска с расставленными ферзями.
	L – логическая переменная, флаг, признак того, что все ферзи расставлены.
	i, k – индексы для работы с массивом.
	d – счетчик для подсчета количества найденных решений.
	f – переменная для вывода данных в текстовый файл.

`Procedure ClearBoard(var p:TBoard);`
`  var i,j:Integer;`
`begin`
`  for i:=1 to n do`
`  for j:=1 to n do`
`  p[i,j]:=0;`
`end;`
	Создадим процедуру для очистки доски.

`Procedure PrintBoard (p: TBoard);`
`  var i,j:integer;`
`begin`
`  for i:=1 to n do`
`  begin`
`  for j:=1 to n do`
`  if p[i,j]=8`
`  then write(f,'[Q]')`
`  else write(f,'[ ]');`
`  writeln(f)`
`  end;`
`  writeln(f);`
`end;`
	Создадим процедуру, которая будет выводить шахматную доску в файл. Будем имитировать клетки доски с помощью квадратных скобок. В соответствующие поля, где будут обнаружены ферзи, будем выводить знак ‘Q’[2].

`Procedure move(var p:Tboard; x,y:integer; L:boolean; k:integer; var d:integer);`	Пишем рекурсивную процедуру. На вход процедура получает доску p с уже расставленными k ферзями и количество уже найденных решений. Локальные переменные: x, y – координаты клетки, куда ставится ферзь.

[2] От английского Queen – королева.

	L - Признак того, что доска не заполнена до конца, не все ходы сделаны. На выходе процедура возвращает доску p и количество найденных решений.
var i,j,k0:integer; q:TBoard;	Объявляем локальные переменные: i, j – индексы для работы с массивом. Переменные для запоминания данных для отката в случае неудачи: k0 – номер поставленного ферзя, q – для запоминания первоначального состояния доски.

begin
if L then
begin
q:=p; k0:=k;
	Если еще можно ставить ферзи, то запоминаем состояние доски.

if k=n
then
begin
L:=false;
p[x,y]:=8;
	Если ферзь, который ставим – последний, то ставим ферзя, устанавливаем флаг L как ложь, то есть ферзи на доске все выставлены[3].

Printboard(p); inc(d) end	Печатаем в файл удачную расстановку ферзей, увеличиваем на 1 счетчик найденных решений.

else
begin
p[x,y]:=8;
	В противном случае (если ферзь не последний) ставим ферзя и…

for i:=1 to n do
if p[i,y]=0
then
p[i,y]:=1;
for i:=1 to n do
if p[x,i]=0
then
p[x,i]:=1;
for i:=-n to n do
if (x+i>0)and
(y+i>0)and(x+i<=n)and(y+i<=n)
then
if
p[x+i,y+i]=0
then
p[x+i,y+i]:=1;
for i:=-n to n do
	…отмечаем клетки (записываем в них единицы), которые бьются этим ферзем по вертикали, по горизонтали, по диагоналям.

[3] Важное замечание. Напоминаем в этом месте ученикам, что для рекурсивной процедуры во избежание зацикливания необходимо предусмотреть ситуацию, когда процедура не будет себя вызывать рекурсивно. В данном случае, если поставлен последний n-й ферзь процедура прекращает вызывать сама себя рекурсивно.

`if (x+i>0)and(y-i>0)and(x+i<=n)and(y-i<=n) then if p[x+i,y-i]=0 then p[x+i,y-i]:=1;`	
`inc(k); for j:=1 to n do if p[k,j]=0 then move(p,k,j,L,k,d);`	Ставим ферзя в строку номер k.
`L:=False; p:=q; k:=k0-1 end; end end end;`	Откатываем доску к предыдущему состоянию. Рекурсивная процедура готова.
`begin k:=1; L:=true; d:=0;`	Тело программы. Начинаем ставить ферзей с первой вертикали (k=1), поднимаем флаг – признак того, что задача пока не решена. Занулили счетчик количества решений.
`assign(f,'h:/Queens.txt'); rewrite(f);`	Ассоциировали текстовый файл f с конкретным местом на диске. Очистили файл или создали его, приготовились к записи данных.
`for i:=1 to n do begin ClearBoard(p); Move(p,1,i,L,k,d) end;`	Теперь пытаемся ставить ферзи, начиная с каждой клетки первой вертикали. Перед каждым вызовом рекурсивной процедуры очищаем доску
`write(f,'Количество решений ',d); close(f); end.`	Выводим в файл количество различных решений и закрываем файл. Задача решена.

Итак, программа создана, отлажена, и теперь можно испытать ее для различных размеров доски.

Вот результат работы программы для n=4.

```
[ ][Q][ ][ ]
[ ][ ][ ][Q]
[Q][ ][ ][ ]
[ ][ ][Q][ ]
```

```
[ ][ ][Q][ ]
[Q][ ][ ][ ]
[ ][ ][ ][Q]
[ ][Q][ ][ ]
```

Количество решений 2

При n=8 (обычная шахматная доска) программа находит 92 решения, вот одно из решений:

```
[Q][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][Q][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][Q]
[ ][ ][ ][ ][ ][Q][ ][ ]
[ ][ ][Q][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][Q][ ]
[ ][Q][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][Q][ ][ ][ ][ ]
```

При n=12 программа находит 14200 решений, вот одно из найденных решений:

```
[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][Q]
[ ][ ][ ][ ][ ][ ][ ][ ][Q][ ][ ]
[ ][ ][ ][ ][ ][ ][Q][ ][ ][ ][ ]
[ ][ ][ ][Q][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][Q][ ][ ][ ][ ][ ][ ][ ][ ]
[Q][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][Q][ ][ ][ ][ ]
[ ][Q][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][ ][ ][Q][ ]
[ ][ ][ ][ ][ ][Q][ ][ ][ ][ ][ ]
[ ][ ][ ][Q][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][Q][ ][ ][ ][ ][ ][ ]
```

При n=14 программа находит 365596 решений, вот одно из найденных решений:

```
[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][Q]
[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][Q][ ]
[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][Q][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][ ][ ][Q][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][ ][Q][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][Q][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][Q][ ][ ][ ][ ][ ][ ]
[ ][Q][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][Q][ ][ ]
[Q][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][Q][ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][Q][ ]
[ ][ ][ ][ ][ ][ ][ ][Q][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][Q][ ][ ][ ][ ][ ][ ][ ]
```

Теперь с помощью построенного рекурсивного алгоритма находим количество различных решений для различных размеров доски[4].

Таблица 2

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Число решений	1	-	-	2	10	4	40	92	352	724	2680	14200	73712	365596	2279184	812945

Список иллюстраций

[4] (2014)

Предметный указатель

Список литературы

1. **Гик Евгений** Математика на шахматной доске [Книга]. - Москва : Мир энциклопедий Аванта +, 2009.

2. Задача о восьми ферзях [В Интернете] // https://ru.wikipedia.org. - 13 1 2014 г.. - 15 9 2014 г.. - https://ru.wikipedia.org/w/index.php?title=%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D0%B2%D0%BE%D1%81%D1%8C%D0%BC%D0%B8_%D1%84%D0%B5%D1%80%D0%B7%D1%8F%D1%85&stable=1.