Лекция 6

Языки и методы программирования

Преподаватель – доцент каф. ИТиМПИ
Кузнецова Е.М.

Содержание

- ***Рекурсия – это такой способ организации вычислительного процесса, при котором процедура или функция в ходе выполнения составляющих ее операторов обращается сама к себе.***

свойства рекурсивных алгоритмов:

- Правильный рекурсивный алгоритм не должен создавать бесконечную последовательность вызовов самого себя.
  - Для этого он обязательно должен содержать *нерекурсивный выход*, т.е. при некоторых исходных данных вычисления в алгоритме должны производиться без вызовов его самого - тривиальный случай.
- Определение сложного случая в терминах более простого.
  - При любых исходных данных нерекурсивный выход должен достигаться за конечное число рекурсивных вызовов. Для этого каждый новый вызов рекурсивного алгоритма должен решать более простую задачу, т.е. рекурсивный алгоритм должен содержать определение некоторого сложного случая в терминах более простого случая.

Рекурсия

- Н.Вирт отмечает, что *"...мощность рекурсии связана с тем, что она позволяет определить бесконечное множество объектов с помощью конечного высказывания*".

- *“... обычно понятие рекурсивных алгоритмов объяснялось на неподходящих примерах, из-за чего возникло широкое распространенное предубеждение против рекурсии в программировании”.*

Пример 1. определение факториала.

$n!=1*2*3*...*n.$

$$n! = \begin{cases} 1, & \text{если } n \le 1, \\ (n-1)! \times n, & \text{если } n > 1. \end{cases}$$

Граничным условием в данном случае является $n<=1$.

```
function Factorial(N:integer): Extended;
begin
 if N<=1 then Factorial:=1
 else Factorial:=Factorial(N-1)*N
end;
```

Пример 2. Определим функцию *K(n)*, которая возвращает количество цифр в заданном натуральном числе *n*:

$$K(n) = \begin{cases} 1, & \text{если } n < 10, \\ K(n/10) + 1, & \text{если } n >= 10. \end{cases}$$

```
function K(N:Longint):Byte;
begin
 if N<10 Then K:=1
 else K:=K(N div 10)+1
end;
```

Пример 3. *Вычислить сумму элементов линейного массива*

- ***сумма равна нулю, если количество элементов равно нулю, и сумме всех предыдущих элементов плюс последний, если количество элементов не равно нулю.***

```
program Rec2;
    Type LinMas = Array[1..100] Of Integer;
    Var A : LinMas;
    I, N : Byte;
{Рекурсивная функция}
function Summa(N : Byte; A: LinMas) : Integer;
begin
    if N = 0 then Summa := 0
    else Summa := A[N] + Summa(N - 1, A)
end;
```

Пример 3. *Вычислить сумму элементов линейного массива*

```
{Основная программа}
begin
    write('Количество элементов массива? '); readln(N);
    randomize;
    for I := 1 to N do
    begin
     A[I] := -10 + random(21); write(A[I] : 4)
    end;
    writeln; writeln('Сумма: ', Summa(N, A))
end.
```

Пример 4. *Определить, является ли заданная строка палиндромом, т.е. читается одинаково слева направо и справа налево.*

- Идея решения заключается в просмотре строки одновременно слева направо и справа налево и сравнении соответствующих символов.
- Граничное условие — строка является палиндромом, если она пустая или состоит из одного символа.

```
program Palindrom;
var S : String;
{Рекурсивная функция}
function Pal(S: String) : Boolean;
begin
    if length(S)<=1 then Pal:=True
    else Pal:= (S[1]=S[length(S)]) and Pal(Copy(S, 2,
    length(S) - 2));
end;
{Основная программа}
begin
    write('Введите строку: '); readln(S);
    if Pal(S) then writeln('Строка является палиндромом')
    else writeln('Строка не является палиндромом')
end.
```

Задача о ходе коня

- задача о нахождении маршрута шахматного коня, проходящего через все поля доски по одному разу.

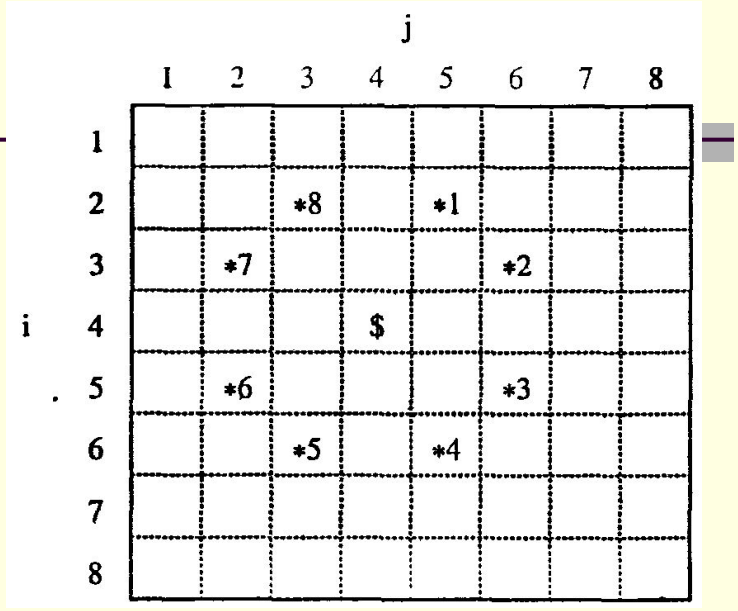
- Эта задача известна по крайней мере с XVIII века. Леонард Эйлер посвятил ей большую работу *«Решение одного любопытного вопроса, который, кажется, не подчиняется никакому исследованию»* (датируется 26 апреля 1757 года).
- Помимо рассмотрения задачи для коня, Эйлер разобрал аналогичные задачи и для других фигур. С тех пор обобщённая задача носит имя «нахождение эйлерова маршрута».

Маршрут Яниша

50	11	24	63	14	37	26	35
23	62	51	12	25	34	15	38
10	49	64	21	40	13	36	27
61	22	9	52	33	28	39	16
48	7	60	1	20	41	54	29
59	4	45	8	53	32	17	42
6	47	2	57	44	19	30	55
3	58	5	46	31	56	43	18

Этот маршрут примечателен во многих отношениях: он образует полумагический квадрат, а при повороте доски на 180° первая половина маршрута (номера с 1 до 32) переходит во вторую (номера с 33 по 64).

- Одной из эвристических стратегий алгоритма может быть следующая. Начиная с произвольного поля i,j (i = 4,j = 4), пытаемся пойти на поле *1, если невозможно, то на поле *2; при неудаче - на поле *3 и т.д. по часовой стрелке



	1	2	3	4	5	6	7	8
1	1			34	3	36	19	22
2			2	37	20	23	4	17
3			33		35	18	21	10
4			38		24	11	16	5
5		32			39	26	9	12
6				25		15	6	27
7	31			40	29	8	13	42
8			30		14	41	28	7

```
Program Tur_Konja;
    var a: array[1..8,1..8] of integer;
    im, jm :array(l..8] of integer;
    i, j, k, n, inac, jnac: integer;
    inext, jnext: integer;
begin {инициализация шахматной доски}
    for i:=1 to 8 do for j:=l to 8 do a[i,j]:=0;
    im[l]:=-2; jm[l]:=l.; im[2]:=-1; jm[2]:=2; im[3]:=1;
    jm[3]:=2; im[4]:=2; jm[4):=l; im[5]:=2; jm[5]:=-1;
    im[6]:=1; jm(6]:=-2; im[7]:=-l; jm[7]:=-2; im[8]:=-2;
    jm[8]:=-l;
    write('введи начальные координаты коня 0<i,j<9: ');
    readln(inac,jnac) ;
    a[inac,jnac]:=1; i:=inac; j:=jnac; n:=2; k:=l;
```

```
    while k<=8 do
    begin inext:=i+im[k]; jnext:=j+jm [k] ;
     if (inext<l) or (inext>8) or (jnext<l) or
     (jnext>8) or (a[inext,jnext]<>0)
     then k:=k+l else
     begin a[inext,jnext]:=n; n:=n+l; i:=inext;
     j:=jnext; k:=l;
    end;
end;
{вывод результата прохода}
    for i:=l to 8 do
    begin writeln; writeln;
     for j:=l to 8 do write(a[i,j]:2,' ')
    end;
    writeln; write('кол-во шагов = ',n-l); readln;
end.
```

В случае отсутствия возможности очередного хода осуществляется возврат коня на предыдущее поле и возобновление поиска дальнейшего маршрута по другому пути. Подобный процесс называют возвратом

```
procedure RETR;
begin
    инициализация начального хода
    repeat выбор очередного хода
     if подходит then его запись;
     if решение не полное then RETR;
     if неудача then стирание хода и возврат на предыдущий until
    удача or нет хода
end.
```

```
program tur;
    var i, j, ii, jj, n, nn: integer; q: boolean;
    dx, dy:array[1..8] of integer; h:array[1..8,1..8] of integer;
{рекурсивная процедура - попытка сделать ход}
    procedure try(i,x,y:integer; var q:boolean);
     var k, u, v: integer; ql: boolean;
    begin
     k:=0; repeat k:=k+l; ql:=false; u:=x+dx[k];    v:=y+dy(k];
     if ( (1<=u) and(u<=n) and (1<=v) and (v<=n) )
    and(h[u,v]=0) then begin h[u,v]:=i;
```

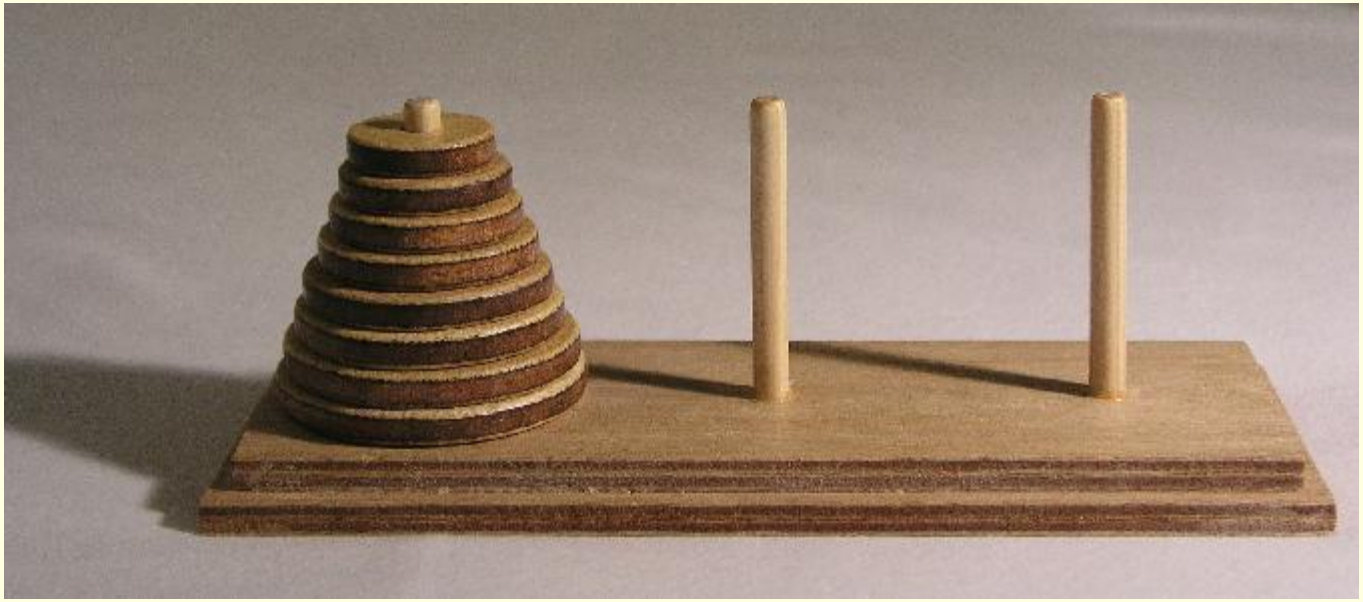
```
{для отладки и наблюдения процесса поиска с возвратом}
        for ii:=l to n do
        begin for jj:= 1 to n do
              write(h[ii,jj]:5); writeln;
        end;
        readin;
        if i<nn then
              begin try(i+l,u,v,ql); if not(ql) then h[u,v]:=0
        else ql:=truer;
    end
   else ql:=true
```

```
    end;
    until (ql) or (k=8);
    q:=ql
end; { конец процедуры}
begin
    dx[l] =2: dx[2]:=l; dx[3]:=-l; dx[4]:=-2; dx[5]:=-2;
    dx[6] =-1: dx[7]:=l; dx[8]:=2; dy[l]:=l; dy[2]:=2;
    dy[3] =2: dy[4]:=l; dy[5]:=-l; dy[6]:=-2;
    dy[7] =-2: dy[8]:=-1;
    write ('введи n: '); readln(n);
```

```
    for i =1 to n do for j:=1 to n do h[i,j]:=0;
    write; ('введи i,j : '); readln(i,j); nn:=n*n;
    h[i,j]:=l; try(2,i,j,q);
    if q then
    begin for i:=l to n do
     begin for j:= 1 to n do write(h[i,j]:5);
           writeln;
     end;
    end;
    else writeln( 'нет маршрута');
    readln
end.
```

Ханойская башня

- *Ханойские Башни* —это головоломка, которую в 1883 г. придумал французский математик *Эдуард Люка*.
- *есть три стержня и восемь дисков разных диаметров, вначале все диски собраны на одном стержне так, что меньшие диски лежат на больших.* Люка предлагал *переложить все диски с первого стержня на третий, используя второй.* При этом следует соблюдать следующее правило: диски можно перекладывать с одного стержня на другой, при этом нельзя класть диск поверх диска меньшего радиуса.



Ханойская башня

- Ради повышения интереса к своей головоломке Люка придумал легенду, повествующую про башню Брамы, увеличенную копию Ханойской. Эта башня состояла то ли из 50, то ли из 64 золотых дисков, а стержни были вырезаны из алмаза. Башни Брамы были созданы при Сотворении мира, и с того времени жрецы в храме трудятся, перекладывая диски.

- для того чтобы перенести самый большой диск, нужно сначала перенести все диски кроме последнего на второй стержень, потом перенести самый большой на третий, после чего останется перенести все остальные диски со второго на третий.

- Задачу о переносе N-1 диска решается аналогично, только поменяем стержни местами (при первом переносе конечным стержнем будем считать второй, а не третий, при втором переносе начальным вместо первого будет второй).
- *Задача о N-1 дисков сводится к задаче о N-2 дисков, та в свою очередь к N-3 дискам, и так вплоть до 1 диска.*

```
program hanoy;
    var n:integer;
 procedure hanoi (n,a,b,c:integer);
 begin
    if n=1 then
     begin
          hanoi (1,a,b,c);
          writeln (a,'->', b);
          exit;
     end
    else
```

```
    begin
      hanoi (n-1,a,c,b);
      hanoi (n-1,c,b,a);
    end;
 end;
begin
    clrscr;
    writeln ('Введите количество колец');
    readln (n);
    hanoi (n,1,2,3);
    writeln ('Нажмите ENTER для выхода');
     readln;
end.
```