Дмитриева Марина Валерьевна

JAVASCRIPT. ЗАНЯТИЕ 4. МЕТОДЫ ПРОГРАММИРОВАНИЯ НА JAVASCRIPT

Средства языка JavaScript позволяют создавать новый HTML-документ в процессе выполнения сценария. Новые, только что созданные документы можно помещать в отдельные окна или заданные фреймы.

Рассмотрим пример создания нового документа при решении задачи построения латинского квадрата. В квадрате размером *N***N* в каждой из клеток требуется поставить одно из чисел 1, 2,...,*N* так, чтобы сумма чисел, стоящих в каждом вертикальном ряду, каждом горизонтальном ряду и по диагонали равнялась одному и тому же числу 1+2+3+...+*N*.

Разобьем окно документа на две горизонтальные области с помощью фреймов. В верхней области поместим условие задачи и форму для ввода размера таблицы. При нажатии на кнопку OK в нижнем фрейме будет помещен документ, содержащий таблицу, представляющую собой латинский квадрат (рисунок 1). Эта таблица строится динамически при выполнении сценария.

Для разбиения окна на две области создается документ, описывающий фреймовую структуру. В данном случае документ имеет простую структуру и может быть задан, например, как в листинге 1 а.

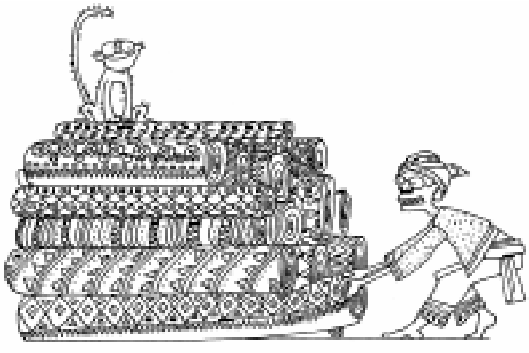
Таблица формируется по строкам. Элемент, который заносится в таблицу с

Рисунок 1. Построение латинского квадрата.

i-ым номером строки и *j*-ым номером столбца, определяется по формуле

`Math.round(Number((j+k-i+3*n-1)%n+1))`,

причем значение *k* зависит от того, четное или нечетное заданное значение *n*. При переходе к формированию следующей строки происходит циклический сдвиг предыдущей.



Листинг 1 а. Фреймовая структура для задачи о латинском квадрате.

```
<HTML>
<HEAD>
<TITLE>Построение латинского квадрата</TITLE>
</HEAD>
<FRAMESET ROWS='35%, *'>
    <FRAME NAME='lattab' SRC='formtab.html'>
    <FRAME NAME='bottom'>
</FRAMESET></HTML>
```

Переменная **out** предназначена для работы с документом, который загружается в нижний фрейм. Это обеспечивается выполнением присваивания **out =top.frames['bottom'].document**. Применением к **out** метода **close()** закрывается поток вывода в документ. Метод **open()** открывает поток вывода в документ. Далее в сценарии формируются элементы в таблице с помощью методов **write()** и **writeln()**, например, так: **out.writeln('<body bgcolor=silver> ')**.

HTML-код документа со сценарием построения латинского квадрата представлен в листинге 1 б.

Листинг 1 б. Построение латинского квадрата.

```
<HTML>
<HEAD>
<title>Построение латинского квадрата</title>
<SCRIPT LANGUAGE='Javascript'>
<!—//
function solvetab()
 {var n = Number(document.forms['form1'].num.value);
  var k
  if (n%2==0)
     k= Number(n/2)
   else
     k= Number((n+1)/2);
   var out = top.frames['bottom'].document;
   out.open()
    out.writeln('<body bgcolor=silver><CENTER>');
   out.writeln('<TABLE bgcolor=white COLS='+n+' ROWS='+n+' BORDER=3>');
   for (i=0; i<n; i++)
     { out.writeln('<TR>');
       for (j=0; j<n; j++)
        {out.writeln('<TD> '+Math.round(Number((j+k-i+3*n-1)%n+1))+' ')}
       out.writeln('</TR>');
      }
    out.write('</TABLE>');
    out.writeln('</CENTER><body>');
   }
-->
</SCRIPT></HEAD>
<BODY>
<i>Построение латинского квадрата</i><BR>
В квадрате размером NxN в каждой из  клеток требуется поставить одно из
чисел  1, 2,..., N так, чтобы сумма чисел в любой строке, столбце и
диагонали равнялась одному и тому же числу 1+2+..+N.<BR>
<FORM NAME='form1'>
Введите N: <INPUT NAME='num' TYPE='text' size=4>
           <INPUT TYPE='button' VALUE='OK' ONCLICK='solvetab()'>
```

Задание.

Магическим квадратом порядка n называется квадратная таблица размером $n*n$, составленная из чисел 1, 2,...,$n2$ так, что суммы по каждому столбцу, каждой строке и каждой из двух диагоналей, равны между собой. Напишите сценарий, который определяет, является ли данная квадратная матрица магическим квадратом.

Можете ли Вы предложить алгоритм построения магического квадрата порядка n?

ОБРАБОТКА ФОРМУЛ

Метод **eval** может быть использован для выполнения операторов языка JavaSvript, включенных в строку параметров. Использование метода **eval** позволяет значительно упростить решение многих задач обработки формул. Часто для представления или преобразования формулы достаточно сформировать строку нужного вида, а затем воспользоваться методом **eval**, который выполнит необходимые преобразования, например, вычислит значение формулы.

На одной из олимпиад по программированию для школьников была предложена следующая задача. Рассматривалась формула вида ((((1?2)?3)?4)?5)?6. Вместо знака ? требовалось поставить одну из арифметических операций так, чтобы результат вычислений был равен 35. Кроме того, требовалось определить количество формул, удовлетворяющих этому условию.

Такую задачу можно решать методом, который получил название метода исчерпывающего или полного перебора. Будем перебирать возможные комбинации

Рисунок 2. Форма для задания значений операндов.

знаков, вычислять значение формулы при каждой комбинации и запоминать те из них, которые удовлетворяют условию.

Сформулируем задачу в общем виде. Напишем функцию, которая для формулы вида ((((*a*?*b*)?*c*)?*d*)?*e*)?*f* находит все комбинации знаков, при которых формула имеет заданное значение *g*. На рисунке 2 изображен документ с формой ввода значений операндов и текстовым полем для размещения найденных формул.

При решении задачи формулу будем хранить в одномерном массиве. Некоторые значения массива нам известны (заданные операнды и скобки), а некоторым элементам следует присвоить значения операций. При задании массива **s** все операции совпадали с операцией плюс. В формуле, представленной массивом, операции являются элементами с индексами 5, 8, 11, 14, 17. Для задания операций используются циклы. Когда очередная комбинация знаков готова, массив преобразуется в строку, применяется метод **eval** для вычисления значения формулы, представленной строкой. Формулы, удовлетворяющие условию, запоминаются в строке **sres**. Описанная функция представлена в листинге 2.

Листинг 2. Построение формул с заданным значением.

```
<HTML>
<HEAD>
<TITLE>Нахождение формул вида  ((((a?b)?c)?d)?e)?f </TITLE>
<SCRIPT LANGUAGE='JavaScript'>
<!— //
function formula (obj)
{var a=Number(obj.data1.value)
 var b=Number(obj.data2.value)
 var c=Number(obj.data3.value)
 var d=Number(obj.data4.value)
 var e=Number(obj.data5.value)
 var f=Number(obj.data6.value)
 var g=Number(obj.data7.value)
 var s=new Array ('(','(','(','(',a,'+',b,')','+',c,')','+',d,')','+',e,')','+',f)
 var op=new Array ('+','-','*','%')
 var r
 var str=''
 var sres=''
 var p=false
 for (var  i=0; i<=3; i++)
  { s[5]=op[i];
    for (var j=0; j<=3; j++)
      { s[8]=op[j];
         for (var k=0; k<=3; k++)
             {  s[11]=op[k];
                for (var m=0; m<=3; m++)
                {  s[14]=op[m];
                    for (var n=0; n<=3; n++)
                        { s[17]=op[n]
                          str=''
                         for (var t=0; t <= s.length-1; t++)
                           {str+=s[t]};
                            r=eval(str)
                            if (r==g)
                               {sres +=str+ '='+g+'\r\n'; p=true}
                         }
                }
             }
       }

  }
if (p)
  obj.result.value=sres
```

```
  else
    obj.result.value='Формул, удовлетворяющих условию, не найдено. '
}
//–>
</SCRIPT>
</HEAD>
<BODY bgcolor=silver >
<h4>Нахождение формул вида ((((a?b)?c)?d)?e)?f</h4>
<FORM name='form1'>
<pre>
a: <INPUT type='text' size=10 name='data1' >
b: <INPUT type='text' size=10 name='data2' >
c: <INPUT type='text' size=10 name='data3' >
d: <INPUT type='text' size=10 name='data4' >
e: <INPUT type='text' size=10 name='data5' >
f: <INPUT type='text' size=10 name='data6' >
g: <INPUT type='text' size=10 name='data7' >
Найденные формулы:
 <textarea cols=30 rows=5 name='result' > </textarea>
</pre><HR>
 <INPUT type='button' value=Определить  onClick='formula(form1)'>
 <INPUT type='reset' value=Отменить>
</FORM></BODY></HTML>
```

Использовать метод **eval** рекомендуется при выполнении упражнения.

Задание.

Напишите сценарий, который позволяет вычислять значения постоянных логических формул, то есть формул, содержащих логические константы **true** и **false**, логическое отрицание, конъюнкцию, дизъюнкцию, импликацию и эквивалентность. Предусмотрите средства для облегчения ввода формул. При нажатии на соответствующую кнопку в текстовое поле помещается требуемое значение так, как показано на рисунке 3. После нажатия на клавишу **Вычислить** вычисляется значение построенной формулы.

РЕКУРСИВНЫЕ ФУНКЦИИ В JAVASCRIPT

При создании программы для решения сложной задачи программист обычно разбивает ее на подзадачи, чтобы для каждой такой подзадачи написать подпрограмму, а затем объединить все написанные подпрограммы в программу. Если подзадача оказывается достаточно сложной, то для ее решения может оказаться полезным разбить ее на еще более мелкие подзадачи и программировать каждую из них отдельно и т. д. Часто в процессе такого разбиения оказывается, что задача сводит-

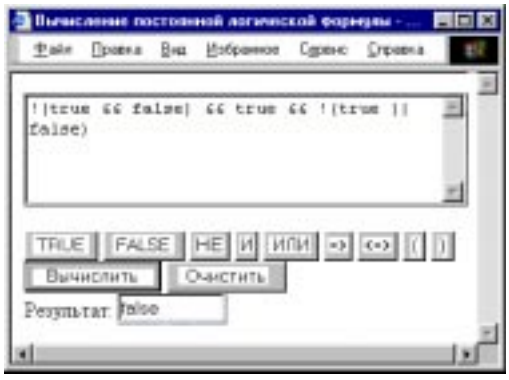
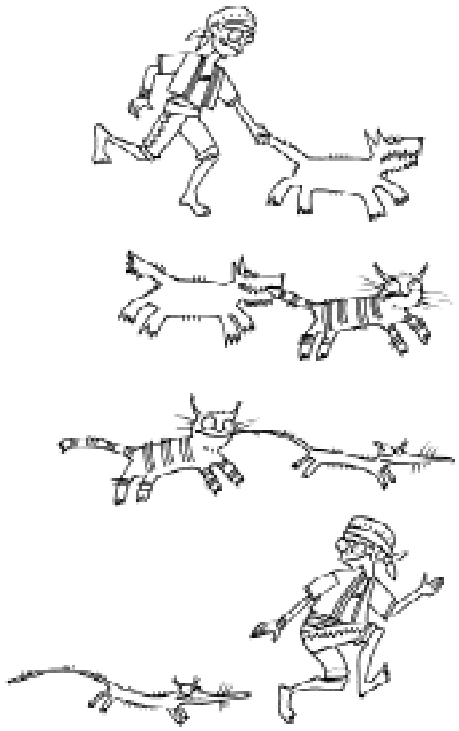


Рис. 3. Построитель логических формул.

ся к самой себе. Если при этом исходные данные оказываются проще, то этот процесс можно продолжать до тех пор, пока исходные данные не окажутся настолько простыми, что решение задачи для них станет тривиальным. Если задача сводится к себе самой, но с другими параметрами, то говорят, что имеет место рекурсия.



Рассмотрим решение задачи, которая получила название задачи о Ханойских башнях. Существует древнеиндийская легенда, согласно которой в городе Бенаресе под куполом главного храма, в том месте, где находится центр Земли, на бронзовой площадке стоят три алмазных стержня. В день сотворения мира на один из этих стержней было надето 64 кольца. Бог поручил жрецам перенести кольца с одного стержня на другой, используя третий в качестве вспомогательного. Жрецы обязаны соблюдать условия:

– переносить за один раз только одно кольцо;

– кольцо можно переносить с одного стержня (A) на другой (B), только если оно имеет меньший диаметр, чем верхнее кольцо, находящееся на стрежне B, или стержень B свободен.

Согласно легенде, когда, соблюдая все условия, жрецы перенесут все 64 кольца, наступит конец света.

Задача состоит в том, чтобы определить последовательность переносов колец минимальной длины.

Язык JavaScript позволяет использовать рекурсивные функции, то есть функции, в теле которых содержится вызов самой функции.

Напишем рекурсивную функцию, которая находит решение задачи о Ханойских башнях для произвольного числа колец. Функция имеет четыре параметра: первый параметр – число переносимых колец, второй параметр – стержень, на который первоначально нанизаны кольца, третий параметр функции – стержень, на который требуется перенести кольца, и, наконец, четвертый параметр – стержень, который разрешено использовать в качестве вспомогательного. Форма для ввода данных изображена на рисунке 4.

Чтобы перенести n колец со стержня A на стержень B, используя стрежень C в качестве вспомогательного, можно поступить следующим образом:

- Перенести $n-1$ кольцо со стержня A на C, используя стержень B в качестве вспомогательного стержня.
- Перенести последнее кольцо со стержня A на стержень B.
- Перенести $n-1$ кольцо со стержня C на B, используя стержень A в качестве вспомогательного стержня.

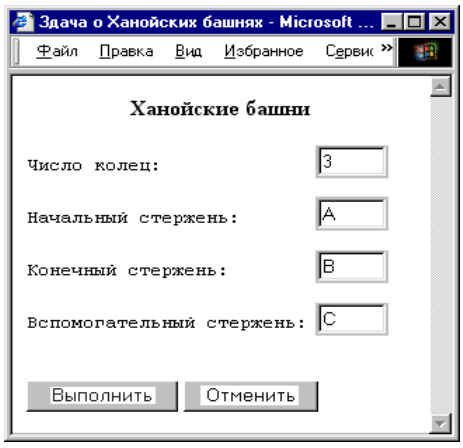


Рисунок 4. Задача о Ханойских башнях.

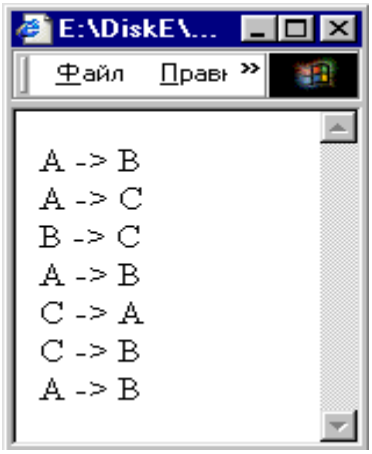


Рисунок 5. Последовательность переноса трех колец.

При перенесении $n-1$ кольца можно воспользоваться тем же алгоритмом, так как на нижнее кольцо с самым большим диаметром можно просто не обращать внимания. Перенос одного кольца в программе выражается в том, что выводится соответствующий ход. HTML-код документа с рекурсивной функцией приведен в листинге 3.

На рисунке 5 приведен результат выполнения сценария при переносе трех колец.

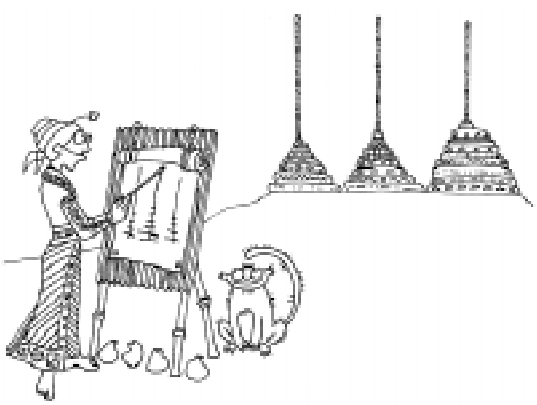
Оценим, сколько переносов будет сделано и сколько шагов выведено при работе программы. Обозначим T_n – число ходов, необходимых для переноса n колец. Очевидно, что $T_0 = 0$, $T_n = 2*T_{n-1} + 1$ при $n > 0$, отсюда получаем $T_n = 2^n - 1$ и $T_{64} = 2^{64} - 1 = 2*10^{19}$.

Если считать, что на перенос одного кольца потребуется одна секунда, то на перенос всех колец потребуется более 5 млрд веков.

Листинг 3. Задача о Ханойских башнях

```
<HTML>
<HEAD>
<TITLE>Здача о Ханойских башнях</TITLE>
<SCRIPT LANGUAGE='JavaScript'>
<!—
function  hb (n,x,y,z)
 { if (n>0)
     { hb (n-1,x,z,y);
       document.writeln ( x,'  ->  ',y,'<br>');
       hb (n-1,z,y,x);
     }
  }
//—>
</SCRIPT>
</HEAD>
<BODY bgcolor=silver>
<h4 align=center>Ханойские башни</h4>
<FORM name='form1'>
<pre>
Число колец:              <INPUT type='text' size=5 name='st1'><HR>
Начальный стержень:       <INPUT type='text' size=5 name='st2'><HR>
Конечный стержень:        <INPUT type='text' size=5 name='st3'><HR>
Вспомогательный стержень: <INPUT type='text' size=5 name='st4'><HR>
</pre>
 <INPUT type='button' value=Выполнить
onClick='hb(form1.st1.value,form1.st2.value,form1.st3.value,form1.st4.value)'>
 <INPUT type='reset' value=Отменить>
</FORM></BODY></HTML>
```

Задание.



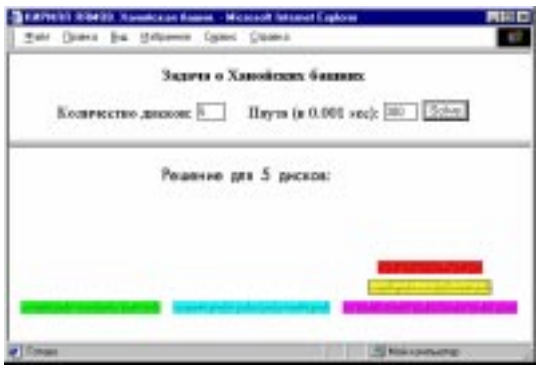


Рисунок 6. Задача о Ханойских башнях с демонстрацией переноса колец.

Напишите сценарий, который демонстрирует работу алгоритма решения задачи о Ханойских башнях. На рисунке 6 приведен один из возможных вариантов решения задачи с демонстрацией переноса колец. Предусмотрена возможность задания скорости перемещения. В любой момент демонстрацию можно прервать, а затем продолжить с прерванной точки.

Дмитриева Марина Валерьевна, доцент кафедры информатики математико-механического факультета Санкт-Петербургского государственного университета.