Дмитриева Марина Валерьевна

JAVASCRIPT. ЗАНЯТИЕ 7. МЕТОД РЕКУРСИВНОГО СПУСКА

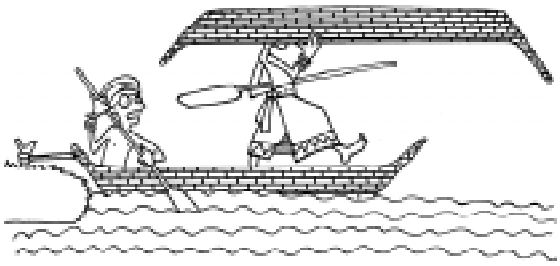
Мой учитель Святослав Сергеевич Лавров многие годы заведовал кафедрой МО ЭВМ СПбГУ и являлся руководителем лаборатории НИИММ (научно-исследовательский институт Математики и Механики), в котором работали и работают в настоящее время его ученики. По инициативе Святослава Сергеевича в университете были начаты работы по разным научным направлениям, подготовлены новые курсы лекций. Святослав Сергеевич впервые в университете прочел курсы лекций «Методы реализации языков искусственного интеллекта», «Автоматическое доказательство теорем», «Теория программирования». Для молодых сотрудников, которыми мы были в то время, Святослав Сергеевич организовал семинары, на которых, в частности, обсуждались вопросы преподавания программирования. Он щедро делился всем тем, что знал и умел. Поражает работоспособность Святослава Сергеевича. Иногда при чтении диссертаций или научном редактировании книг учеников объем его замечаний, пояснений, дополнений был соизмерим с размером рукописи. Я очень благодарна Святославу Сергеевичу за поддержку и в годы учебы, и во время работы.

Во время нашего сотрудничества важное место в программировании занимала теория трансляции. Многие ученики С.С. Лаврова разрабатывали трансляторы с разных языков для различных ЭВМ.

Одним из методов, применяемых при лексическом и синтаксическом анализе программ, являлся метод рекурсивного спуска, который полезен при решении других задач.

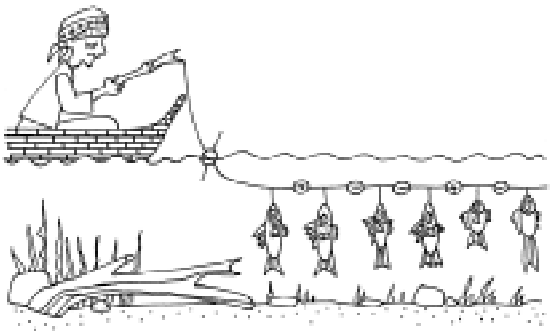
В сети Интернет проводятся разные опросы, данные которых следует обработать с помощью формул и представить в соответствующем виде. Если через Интернет делаются заказы на товары или услуги, то также требуется обработка введенных значений, выполнение вычислений.

При написании программы для сложной задачи часто задачу разбивают на подзадачи, для каждой из которых пишется своя подпрограмма. В дальнейшем подпрограммы следует объединить в целую программу. Если подзадача оказывается сложной, то и ее, в свою очередь, разбивают на подзадачи и т. д. При таком разбиении может возникнуть ситуация, когда задача сводится к себе самой, но с другими, более «простыми», параметрами. Решение такого рода задач можно осуществить рекурсивными методами. Использование рекурсии предпочтительно в случаях, когда данные имеют рекурсивную структуру.



Продемонстрируем метод рекурсивного спуска при решении задачи вычисления значения формулы. Напишем сценарий, вычисляющий значение такой формулы. Для определенности будем считать, что формула состоит из целочисленных констант (от 0 до 9), знаков операций и круглых скобок. Допустимыми операциями являются сложение, вычитание, умножение. Формулу при решении задачи требуется просмотреть слева направо один раз.

Для решения задачи удобно ввести следующие понятия, характеризующие отдельные части формулы (подформулы) и формулу целиком. Назовем *формулой* один терм, либо последовательность термов,



соединенных знаками «+» или «–». Будем называть *термом* либо один множитель, либо последовательность множителей, соединенных знаками «*». Назовем *множителем* либо константу (от 0 до 9), либо формулу, заключенную в скобки.

При анализе формулы будем использовать функцию **cursym()**, которая выдает текущий символ рассматриваемой формулы.

Опишем функцию **valform()**, которая анализирует формулу и, если формула построена правильно, вычисляет ее значение. По данному определению, формула всегда начинается с терма. Будем считать, что функция **valterm()** вычисляет значение для подформулы, которую мы определили как терм. После вызова функции **valterm()** будет вычислено значение первого терма. Если далее в формуле следует знак плюс или знак минус, то за знаком в формуле, согласно нашему определению, должен опять следовать терм. Вычислим значение и этого терма, воспользовавшись процедурой **valterm()**, далее можно вычислить значение подформулы с двумя термами. Процесс продолжается до тех пор, пока после анализа очередного терма текущий символ станет отличным от знака плюс или знака минус. Описание функции **valform** представлено в листинге 1.

Напомним, что при вычислении значения формулы мы воспользовались функцией вычисления значения терма. При описании функции опираемся на определение терма. Терм состоит либо из одного множителя, либо из множителей, соединенных знаком умножения. Поступим так, как поступили при описании функции **valform()**. Будем считать, что функция **valmn()** вычисляет значение формулы, определенной как множитель. Для того чтобы вычислить значение терма, следует сначала вычислить значение множителя. Анализируя знак операции, следующий за первым термом, мы либо прекращаем анализ формулы, либо вычисляем значение следующего множителя

Листинг 1. Вычисление значения формулы

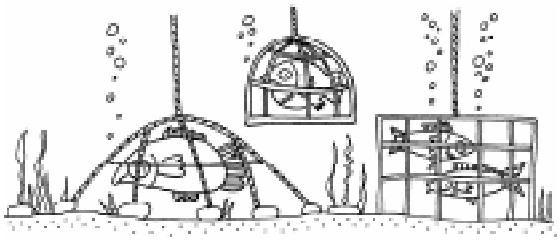
```
function  valform ()
 { var t1 = valterm ()
   var t2
   while ((c == "+")|| (c == "-"))
   { var h = c
     c =cursym ()
     t2 = valterm ()
     if (h=="+") t1 = t1 + t2
     else t1= t1 - t2
   }
  return Number(t1)
 }
```

Листинг 2. Функция вычисления значения терма

```
function  valterm ()
{ var t1 = valmn ()
  var t2
  while (c == "*")
    { c = cursym()
      t2 = valmn()
      t1 = t1 * t2
    }
  return Number(t1)
 }
```

и т. д. Функция **valterm()** очень похожа на функцию **valform()**, ее описание дано в листинге 2.

Далее следует описать функцию **valmn()**. Множителем может быть константа, ее значение нам известно. Множителем может быть формула в скобках, то есть множитель мо-



жет иметь вид **(F)**. Если же очередной символ не является ни числом, ни открывающей скобкой, то формула содержит ошибку.

Рассмотрим случай, когда множителем является формула в скобках. Значение формулы **(F)** совпадает со значением формулы **F**, а значение формулы мы уже умеем вычислять с помощью функции **valform()**, которой мы и воспользуемся. После вычисления значения формулы **F** очередным символом должна быть закрывающая скобка. Если это не так, то нарушен баланс скобок и формула содержит ошибку.

Будем использовать логическую переменную **er**, значение которой изменится на **true** в случае определения ошибки в формуле. Приведем в листинге 3 описание функции **valmn()**.

Заметим, что функция **valform()** содержит вызов функции **valterm()**, которая содержит вызов функции **valmn()**, последняя, в свою очередь, содержит вызов **valform()**. При анализе формулы осуществлен рекурсивный спуск.

Напомним, что рассмотренная нами ранее функция **eval(s)** рассматривает строку **s** как выражение и вычисляет ее значение.

В программе для контроля работы описанных функций используется функция **valtest()**,

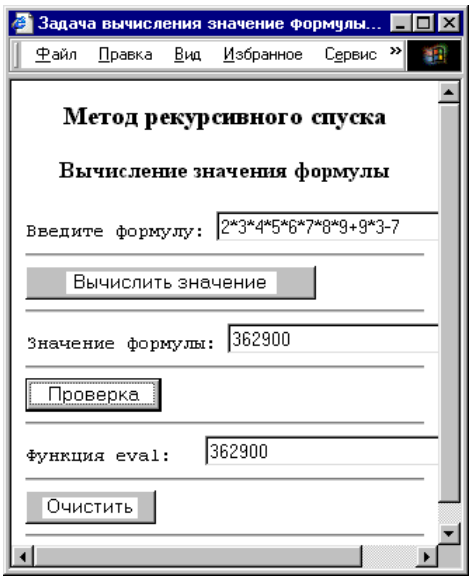
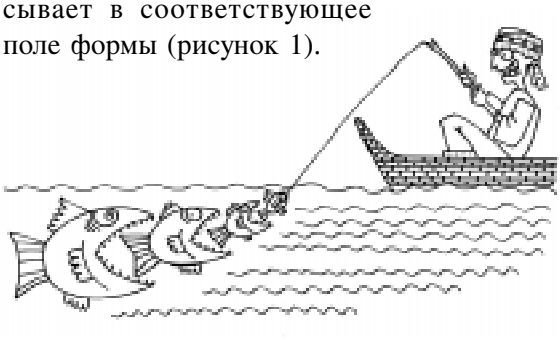


Рисунок 1. Пример работы сценария вычисления значения формулы.

которая обращается к функции **eval** и вычисленное с ее помощью значение записывает в соответствующее поле формы (рисунок 1).



Листинг 3. Вычисление значения множителя

```
function  valmn ()
 { if  ((c >= "1") && ( c <= "9"))
  { var h=Number (c)
    c = cursym ();
    return h
  }
 else
  { if (c == "(" )
     { c = cursym()
       var t = valform()
        if (c != ")")   er= true
       else
         { c = cursym(); return Number(t) }
     }
   else  er = true
  }
```

Функция **eval** обладает большими возможностями, чем приведенная программа. В этом смысле программа носит демонстрационный характер. Но если в формуле используются знаки операций, не определенные в языке JavaScript, то воспользоваться методом **eval** не удается, требуется писать свою программу определения значения формулы. Основные функции написаны, полностью создать документ читателю представляется в качестве упражнения.

Можно облегчить ввод формул, используя Построитель формул, представленный на рисунке 2. В зависимости от того, из каких операндов и операций состоят формулы, пользователь может описать свой Построитель формул.

Листинг 4. Построитель формул

```
<HTML> <HEAD> <TITLE>Построитель формул </TITLE>
<script>
<!--
var s=""
//функция вычисляет значение формулы
function val(obj)
{ obj.m1.value=s;   obj.res.value=eval(s)}
//функция добавляет символ к формуле
function valbut (c)
{s+=c; document.form1.m1.value=s}
//-->
</script></HEAD>
<body bgcolor="#FFDCD">
<Center>
<h4>Построитель формул</h4>
<table border=0 cellspacing=5 cellpadding=5>
<tr valign=top>
 <td align=center >
  <form name ="form1">
   <textarea name="m1" cols=30 rows=3></textarea></td></tr>
<tr  valign=middle><td >
 <input type=button value=1 onclick="valbut('1')">
 <input type=button value=2 onclick="valbut('2')">
 <input type=button value=3 onclick="valbut('3')">
 <input type=button value=4 onclick="valbut('4')">
 <input type=button value=5 onclick="valbut('5')">
 <input type=button value=6 onclick="valbut('6')">
 <input type=button value=7 onclick="valbut('7')">
 <input type=button value=8 onclick="valbut('8')">
 <input type=button value=9 onclick="valbut('9')">
 <input type=button value=0 onclick="valbut('0')">
 <input type=button value=+ onclick="valbut('+')">
 <input type=button value=- onclick="valbut('-')">
 <input type=button value=* onclick="valbut('*')">
 <input type=button value=/ onclick="valbut('/')">
 <input type=button value=( onclick="valbut('(')">
 <input type=button value=) onclick="valbut(')')"><br>
 <input type=button value=Вычислить onclick="val(form1)"><br>
Значение:<input type=text   name="res"  size=20><br>
<input type=reset value=Отменить onclick="s='" >
</form></td></tr></BODY></HTML>
```

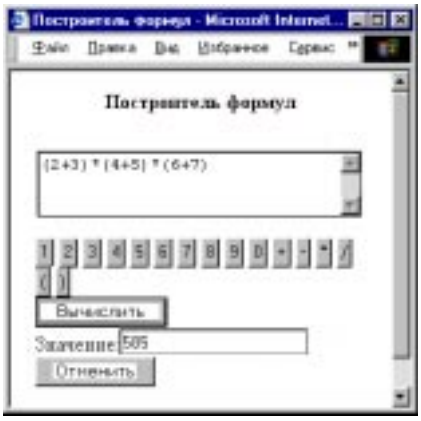


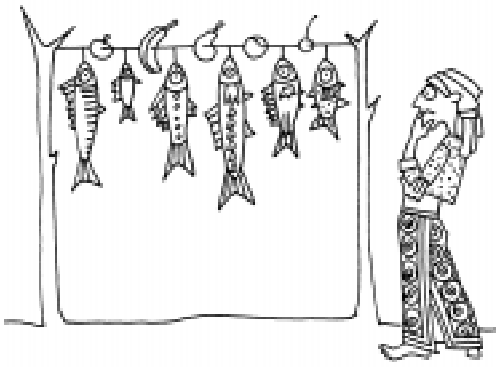
Рисунок 2. Построитель формул.

В листинге 4 приведен HTML-код документа с построителем формул.

Описанный метод рекурсивного спуска оказывается полезным при решении многих задач. Для решения задач, предложенных в качестве упражнений, рекомендуется использовать метод рекурсивного спуска.

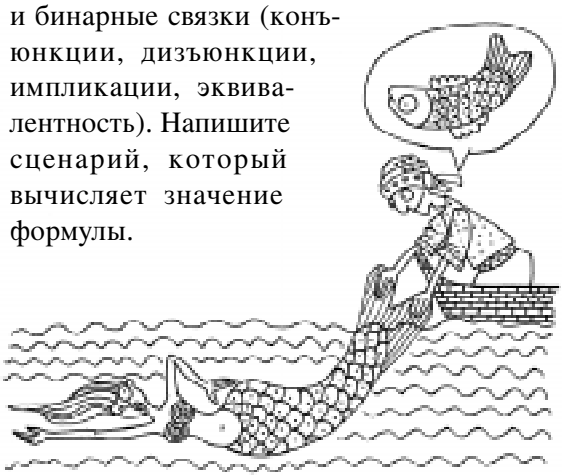
Задания.

1. Напишите функцию, которая по формуле, содержащей однобуквенные идентификаторы, круглые скобки, знаки операций сложения, вычитания, умножения, определяет, правильна ли построена формула.



2. Пусть формула строится из логических констант, круглых скобок, логического отрицания, логического И, логического ИЛИ. Напишите сценарий, который вычисляет значение логической формулы. Для облегчения ввода формулы создайте «построитель формулы».

3. Рассмотрим пропозициональные формулы, содержащие логические константы, круглые скобки. В формуле используется знак логического отрицания и бинарные связки (конъюнкции, дизъюнкции, импликации, эквивалентность). Напишите сценарий, который вычисляет значение формулы.



4. Напишите сценарий, вычисляющий значение формулы, содержащей целые константы, операции +, –, *, / и представленной в префиксной нотации (прямая польская запись). Напомним, что формула в префиксной нотации выглядит так $\oplus AB$, где $\oplus$ – знак операции, A и B – формулы в префиксной нотации. Формула $a + b*c$ в префиксной нотации запишется так $+a*bc$, а формула $(a + b)*c – (d + e)/k$ следующим образом: $–*+abc/+dek$. Скобки при таком представлении формулы не требуются, так как для каждой операции известны операнды. Формула состоит из переменных (представленных однобуквенными идентификаторами), знаков операций: плюс, минус, умножение и деление.

5. Напишите сценарий, который по произвольной формуле в инфиксной нотации строит формулу в постфиксной нотации. Напомним, что при записи формулы в постфиксной нотации (обратной польской записи) знак операции ставится непосредственно после операндов. Формула в постфиксной нотации имеет вид $AB\oplus$, где $\oplus$ –

знак операции, *A* и *B* – формулы в постфиксной нотации. Формула состоит из переменных (представленных однобуквенными идентификаторами), знаков операций: плюс, минус, умножение и деление.

6. Напишите сценарий, который определяет полное сопротивление параллельно-последовательной (П.-П.) схемы. Формально, понятие П.-П. схемы можно представить следующим образом:

• Один резистор является П.-П. схемой. Сопротивление схемы равно сопротивлению единственного входящего в нее резистора.

• Последовательное соединение П.-П. схем является П.-П. схемой. Сопротивление схемы равно сумме сопротивлений последовательно соединенных схем.

• Параллельное соединение П.-П. схем является П.-П. схемой. Сопротивление схемы вычисляется по формуле

$$R = \frac{1}{\frac{1}{R_1} + \ldots + \frac{1}{R_n}}$$

При решении задачи рекомендуется изображение электрической схемы записать по определенным правилам в виде строки символов (формулы). Если схема состоит из одного проводника, то задается число – величина сопротивления. Последовательное соединение схем можно задать строкой вида $(A_1 + A_2 + \ldots + A_n)$, где A_i – либо один проводник, либо электрическая схема. Параллельное соединение можно задать строкой вида $(A_1 * A_2 * \ldots * A_n)$, где A_i – либо один проводник, либо электрическая схема.

© Наши авторы, 2003.
Our authors, 2003.

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