3.6. ОПЕРАТОРЫ УПРАВЛЕНИЯ ВЫЧИСЛИТЕЛЬНЫМ ПРОЦЕССОМ

Содержание

3.6.1. Оператор if

3.6.2. Условный оператор

3.6.3. Операторы организации цикла

3.6.4. Метки и операторы: continue, break, switch

3.6.5. Примеры типовых алгоритмов

3.6.6. Упражнения

3.6.7. Контрольные вопросы

3.6.8. Источники

3.6.1. Оператор if

Оператор **`if`** или **оператор организации ветвлений**, позволяет выбрать и запустить на выполнение одну из альтернативных групп операторов. Выбор осуществляется с помощью булевых значений (`true` или `false`). Рассмотрим, следующий пример использования оператора `if`.

Листинг 3.6.1

```
<html> <head><title>Пример 3.6.1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
<h2>Пример 3.6.1. Оператор if с использованием метода confirm()</h2>
<script type="text/javascript">
var response =
confirm("Вы знаете Javascript? - Нажмите OK. А если нет - Отмена.");
if ( response == true)
  {
    var str = "Отлично! Знание Javascripta — то, что необходимо"
    alert(str)
  }
  else
  {
    var str = "Жаль! Javascript надо бы знать."
    alert(str)
  }
if (response) document.write("<h3>Результат опроса: " + str + "</h3>")
if (!response) document.write("<h3>Результат опроса: " + str + "</h3>")
</script>
</body> </html>
```

В этом примере, когда вы щелкаете на кнопке OK, метод `confirm()` создаёт булево значение `true`, которое присваивается переменной `response`. Переменная `response` в операторе `if` проверяется на равенство значению `true`. Проверка на равенство выдаёт `true`, поэтому выполняются два оператора , стоящие после условия в фигурных скобках и в результате на экран с помощью метода `alert()` выводится `Отлично! … .` Если же вы нажимаете на `Отмена`, создается и запоминается в переменной `response` булево значение `false`. В этом случае срабатывают операторы, стоящие в фигурных скобках после `else`.

Фигурные скобки «**{**» и «**}**» называются **операторными скобками.** Операторные

скобки используются **для объединения операторов в группы**. Легко усвоить смысл фигурных скобок, если вспомнить использование скобок для выделения многочленов в школьной алгебре.

Синтаксис оператора **if** можно записать следующим образом:

if (условие) {операторы1;}else{операторы2;}.

Работу оператора **if** можно описать так: **если условие истинно (true), то выполняются операторы1, если условие ложно (false), то выполняются операторы2.**

Можно использовать и сокращённый вариант оператора if:

if (условие) {операторы1;}.

Работу сокращённого оператора **if** можно описать так: **если условие истинно (true), то выполняются операторы1, если условие ложно (false), то начинают выполняться операторы, стоящие после оператора if**.

В примере 3.6.1 есть два сокращённых оператора **if**, выводящих результат опроса с помощью метода `document.write()`. В качестве условия в этих операторах используется непосредственно переменная `response` без использования операции сравнения, поскольку в этой переменной и без того находится логическое значение `true` или `false`. На примере этих двух операторов можно также понять, что, если после условия стоит один оператор, то операторные скобки можно не использовать. То же самое справедливо для оператора, стоящего после `else`. Например, два усечённых оператора в листинге 3.6.1 можно заменить одним:

```
if (response) document.write("<h3>Результат опроса: "+str+"</h3>")
else document.write("<h3>Результат опроса: " + str + "</h3>")
```

3.6.2. Условный оператор

Условный оператор может заменить оператор if в наиболее простых случаях, когда в качестве альтернатив используется по одному оператору . Например, операторы:

```
x=prompt('ведите целое число',1);
if (x>0) y=Math.log(x); else y=Math.log(Math.abs(x));
```

можно заменить операторами:

```
x=prompt('ведите целое число',1);
y=(x>0) ? Math.log(x) : Math.log(Math.abs(x));
```

Можете убедиться, что результаты будут одинаковыми.

Синтаксис условного оператора можно записать в виде:
условие ? оператор1 : оператор2

В результате выполнения условного оператора срабатывает один из двух вложенных в него операторов в зависимости от значения условия: если условие истинно, то выполняется первый оператор, если ложно — второй.

Синтаксис второго варианта условного оператора можно записать так:
переменная = условие ? выражение1 : выражение2

В этом варианте записи условный оператор возвращает результат вычисления одного из выражений в зависимости от значения условия: если условие истинно, то вычисляется и присваивается переменной первое выражение, если ложно — второе.

В листинге 3.6.2 приводятся три примера использования условного оператора, которые заменяют один и тот же оператор `if`.

Листинг 3.6.2.

```
<html> <head><title>Пример 3.6.2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
<h2>Пример 3.6.2. Оператор <b>if</b> можно заменить условным
оператором</h2>
<script type="text/javascript">
var str;
var str1 = "Отлично! Знание Javascripta — то, что необходимо"
var str2 = "Жаль! Javascript надо бы знать."
var response = confirm("Вы знаете Javascript? - Нажмите OK. А если нет -
Отмена.");
// оператор if
if (response)
document.write("<h3>Результат опроса (оператор if) : " + str1 + "</h3>")
else
document.write("<h3>Результат опроса (оператор if): " + str2 + "</h3>");
// условный оператор (пример 1)
response ?
document.write
("<h3>Результат опроса (условный оператор1):"+str1+"</h3>") :
document.write
("<h3>Результат опроса (условный оператор1): "+str2+ "</h3>");
// условный оператор (пример 2)
response ? str=str1 : str=str2;
document.write("<h3>Результат опроса (условный оператор2): "+str+"</h3>")
// условный оператор (пример 3)
str = response ? str1 : str2;
document.write("<h3>Результат опроса (условный оператор3): "+str+"</h3>")
</script>
</body>
</html>
```

Повторите этот пример, чтобы убедиться, что вы освоили использование условного оператора.

3.6.3. Операторы организации цикла

Цикл – это повторение последовательности операторов некоторое количество раз до тех пор, пока выполняется некоторое условие. В языке JavaScript существует четыре оператора цикла:

- **do … while**
- **while**
- **for**
- **for … in**

Рассмотрим первые два оператора – операторы: **do … while** и **while**. Оператор цикла **while** называют оператором цикла с предусловием, а оператор организации цикла **do...while** - оператором цикла с постусловием. Их различие следует из названий.

Оператор do…while запускает на выполнение оператор или группу операторов и повторяет эту процедуру до тех пор, пока условие не перестанет выполняться. Синтаксис этого оператора можно записать так:

do {группа операторов} while (условие)

Группа операторов заключается в **операторные скобки** (в языке Javascript – это

фигурные скобки). Если в группу входит лишь один оператор, то **операторные скобки** можно не использовать. Эти операторы, заключённые в операторные скобки (или не заключённые, если лишь один оператор), часто называют **телом цикла**. Понятие “тело цикла” будем использовать во всех операторах цикла.

Это определение поясним на следующем примере.

Листинг 3.6.3.

```
<html>
<head><title>Пример 3.6.3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
<h2>Пример 3.6.3. Оператор <b>do...while</b> - оператор цикла с
постусловием</h2>
<script type="text/javascript">
var x = 0
do
  {
    x=x+1;
    alert(x);
  }
while (x < 4);
</script>
</body>
</html>
```

В данном сценарии имеется переменная **x**, которой до цикла присваивается значение, равное нулю. Затем в теле цикла (после **do)** первый оператор увеличивает значение **x** на **1,** а второй выводит результат в окне предупредительных сообщений. В условии (после **while)** указывается, что операторы тела цикла должны повторяться до тех пор, пока значение **x** меньше **4**.

Необходимо заметить, что, если условие с самого начала не выполняется, то операторы, стоящие в теле цикла всё равно один раз выполнятся, поскольку условие проверяется после того, как выполнятся операторы тела цикла (поэтому и название оператора — оператор цикла с постусловием).

В качестве одного из операторов в теле цикла обязательно должен быть оператор, меняющий переменную, входящую в условие так, чтобы рано или поздно условие перестало выполняться (в примере это `x=x+1)`. Иначе цикл никогда не завершится и программа «зависнет».

Оператор while подобен оператору `do … while` и действует похожим образом, только условие, при котором выполнятся (или не выполнятся) операторы тела цикла, стоит в самом начале (поэтому — оператор с предусловием). В операторе `while`, в отличие от оператора `do … while`, операторы тела цикла могут вовсе не запускаться на выполнение, если условие с самого начала не выполняется. Синтаксис оператора while имеет вид:

while (условие) {группа операторов}

Это определение реализовано в следующем примере.

Листинг 3.6.4.

```
<html>
<head><title>Пример 3.6.4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
```

```
<h2>Пример 3.6.4. Оператор <b>while</b> - оператор цикла с
предусловием</h2>
<script type="text/javascript">
var x = 0
while (x < 4)
  {
    x=x+1;
    alert(x);
  }
</script>
</body>
</html>
```

Оператор for называется оператором цикла с переменной-параметром. Часто этот оператор называют также оператором цикла со счётчиком. Синтаксис оператора можно записать в виде:

```
for (присвоение_начального_значения_переменной-параметру;
    условие;
    присвоение_следующего_значения_переменной-параметру)
    {группа_операторов}
```

Цикл `for` выполняется до тех пор, пока условие истинно (принимает значение `true`).

Всё, что указывается в круглых скобках после ключевого слова **`for`** часто называют **“шапкой” цикла**.

В шапке цикла один-единственный раз, в самом начале выполнения оператора `for`, с помощью оператора присваивания переменной-параметру (или переменной-счётчику) необходимо присвоить начальное значение. В качестве переменной-параметра может быть использована любая переменная.

Затем вычисляется условие - операция сравнения или логическое выражение, которое вычисляется каждый раз перед выполнением тела цикла. Операторы тела цикла могут ни разу не выполниться, если условие с самого начала ложно (равно false).

Значение переменной-параметра изменяется каждый раз после очередного прохода цикла, то есть уже после того, как выполнятся операторы тела цикла.

Предупреждение! При неправильном задании условия цикл может повторяться бесконечно, как, например, здесь: `for(x=1;x>0;x++)`.

Чтобы усвоить определение оператора for необходимо выполнить следующий пример.

Листинг 3.6.5.

```
<html>
<head><title>Пример 3.6.5</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
<h2>Пример 3.6.5. Оператор <b>for</b> - оператор цикла с переменной-
параметром</h2>
<script type="text/javascript">
var x = 0;
for (var i = 0; i < 4; i=i+1)
  {
    x=x+1;
    alert(x);
  }
alert ('переменная-параметр после выхода из цикла = ' + i)
```

```
</script>
</body>
</html>
```

В листинге 3.6.5 «шапка» цикла - `for(var i=0;i<4;i=i+1)`. Значение переменной **`i`** будет увеличиваться от нуля с шагом, равным единице, до тех пор, пока оно не станет равным 4. Каждый раз после проверки условия, пока это условие истинно, выполняются операторы тела цикла: `{x=x+1;alert(x);}`. Как только условие перестаёт выполняться, осуществляется переход к операторам, стоящим после оператора цикла. В примере 3.6.5 после цикла стоит оператор, выводящий на экран значение переменной цикла, которое сделало ложным значение условия и привело к завершению оператор цикла.

Оператор `for` удобен при работе с массивами. Вспомним пример листинга 3.4.1 из раздела 3.4 и сравним его со следующим примером из листинга 3.6.6.

Листинг 3.6.6.

```
<html>
<head><title>Пример 3.6.6. Цикл for и одномерный массив</title>
<META http-equiv=Content-Type content="text/html; charset=UTF-8" />
</head>
<body>
<h2>Пример 3.6.6. Использование цикла for для работы с одномерным массивом</h2>
<script>
//
// создаём массив, содержащий дни недели
//
var days_of_week_russ = new Array(7);
days_of_week_russ[0]="Понедельник";
days_of_week_russ[1]="Вторник";
days_of_week_russ[2]="Среда";
days_of_week_russ[3]="Четверг";
days_of_week_russ[4]="Пятница";
days_of_week_russ[5]="Суббота";
days_of_week_russ[6]="Воскресенье";
//
// выводим значения элементов массива
//
for (var i=0; i<days_of_week_russ.length; i++) {
    document.write("<p>");
    document.write(days_of_week_russ[i]);
    document.write("</p>");
}
</script>
</body>
</html>
```

В примере 3.6.6 массив создаётся по-прежнему «вручную». А вывод массива осуществляется с использованием цикла `for`. Нетрудно представить себе насколько сокращается трудоёмкость программирования по сравнению с программированием «вручную» при выводе значений массива, имеющего большую длину.

Ещё более эффективно использование цикла for для работы с многомерными массивами. Пример работы с двумерным массивом из листинга 3.4.2 раздела 3.4 можно переписать следующим образом:

Листинг 3.6.7.

```
<html><head>
<title>Пример 3.6.7. Цикл for и двумерные массивы.</title>
<META http-equiv=Content-Type content="text/html; charset=UTF-8" />
<link rel="stylesheet" type="text/css" href="Petrov.css" />
<style type="text/css">
table {border: 1px solid black; border-collapse:collapse}
td {border: 1px solid gray;}
</style
</head>
<body>
<h2>Пример 3.6.7. Использование оператора цикла for<br/>
для работы с двумерными массивами.</h2>
<script type="text/javascript">
var a=-50.00, b=50.00;
var n=3, m=2;
var i,j;
// создаём пустой двумерный массив размером n*m
var arr_2 = new Array(n);
for (var i=0; i<n; i++)
    arr_2[i]=new Array(m);
// заполняем массив случайными числами
for (i=0; i<n; i++)
    for (j=0; j<m; j++)
        arr_2[i][j]=a+(b-a)*Math.random();
// выводим массив в виде таблицы
// вначале выводим "шапку" таблицы с номерами столбцов
document.write("<table>");
document.write("<tr><td></td>");
for (j=0; j<m; j++) {
    document.write("<td>Столбец"+j+"</td>");
}
document.write("</tr>");
// затем - номера строк и строку со значениями массива
for (i=0; i<n; i++){
    document.write("<tr><td>Строка"+i+"</td>");
    for (j=0; j<m; j++)
        document.write("<td>"+arr_2[i][j].toFixed(2)+"</td>");
    document.write("</tr>");
}
document.write("</table>");
</script></body></html>
```

В комментариях листинга 3.6.7 даны пояснения. Дополним комментарии разъяснениями относительно так называемых вложенных циклов. Первый такой цикл используется для заполнения массива случайными числами:

```
// заполняем массив случайными числами
for (i=0; i<n; i++)
    for (j=0; j<m; j++)
        arr_2[i][j]=a+(b-a)*Math.random();
```

В этом фрагменте в тело внешнего оператора цикла с «шапкой» `for (i=0; i<n; i++)` вложен один оператор, который также является оператором цикла: `for(j=0;j<m; j++) arr_2[i][j]=a+(b-a)*Math.random();`.

Возможно вы забыли, что означают операторы `i++` и `j++`, используемые в шапке цикла. Это операторы инкремента. Эти операторы аналогичны операторам `i=i+1` и `j=j+1`.

Оператор `for … in` – последний, рассматриваемый нами оператор цикла. Синтаксис оператора:

`for (переменная in имя_массива) {группа_операторов}`

Группа операторов, входящая в тело цикла, будет выполнена столько раз, сколько элементов имеет массив, указанный, после ключевого слова **`in`**. При этом переменной, указанной перед `in`, будет последовательно присваиваться значение, содержащее индекс текущего элемента массива. Пример 3.6.6 с использованием оператора for...in будет выглядеть следующим образом:

Листинг 3.6.8.

```
<html>
<head><title>Пример 3.6.8. Цикл for...in и одномерный массив</title>
<META http-equiv=Content-Type content="text/html; charset=UTF-8" />
</head>
<body>
<h2>Пример 3.6.8. Использование цикла for...in для работы с одномерным массивом</h2>
<script type="text/javascript">
//
// создаём массив, содержащий дни недели
//
var days_of_week_russ = new Array(7);
days_of_week_russ[0]="Понедельник";
days_of_week_russ[1]="Вторник";
days_of_week_russ[2]="Среда";
days_of_week_russ[3]="Четверг";
days_of_week_russ[4]="Пятница";
days_of_week_russ[5]="Суббота";
days_of_week_russ[6]="Воскресенье";
//
// выводим значения элементов массива
//
for (i in days_of_week_russ) {
    document.write("<p>");
    document.write(days_of_week_russ[i]);
    document.write("</p>");
}
</script>
</body>
</html>
```

Более эффективно использование оператора `for...in` для массивов, имеющих неупорядоченные или текстовые индексы (вспомним: массивы, имеющие текстовые индексы называют **ассоциативными массивами**).

Следующий пример иллюстрирует создание документа, содержащего расписание занятий по дисциплине. В качестве индексов используются наименования дней недели, а значениями являются строки, содержащие расписание. Заполняется расписание по-прежнему «вручную», а вывод автоматизирован с использованием оператора `for...in`.

Листинг 3.6.9.

```
<html><head>
<title>
Пример 3.6.9. Цикл for...in и одномерный массив с текстовыми индексами
</title>
</head>
<body>
<h2>Пример 3.6.9. Использование цикла for...in для работы<br/>
```

```
с одномерным массивом, имеющим текстовые индексы</h2>
<h3>Расписание занятий по дисциплине "ВТ и программирование"</h3>
<script type="text/javascript">
//
// создаём массив, содержащим расписание занятий по дням недели
//
var days_of_week = new Array(7);
days_of_week["Понедельник"]="1-я и 2-я пары. Лекция.";
days_of_week["Вторник"]="3-я и 4-я пары. Лабораторные работы";
days_of_week["Среда"]="6-я и 7-я пары. Самостоятельная работа";
days_of_week["Четверг"]="Нет занятий по \"ВТ и программирование\"";
days_of_week["Пятница"]="6-я пара. Консультации.";
days_of_week["Суббота"]="Свободный график. Самостоятельная работа";
days_of_week["Воскресенье"]="Выходной";
//
// выводим индексы и значения элементов массива
//
for (i in days_of_week) {
    document.write("<p>");
    document.write(i+" : "+days_of_week[i]);
    document.write("</p>");
}
</script></body></html>
```

Благодаря строке `for (i in days_of_week)` переменная `i` последовательно принимает значения, соответствующие индексам элементов массива и используется в теле цикла для вывода этого индекса и значения элемента массива, соответствующего этому индексу.

3.6.4. Метки и операторы: `continue`, `break`, `switch`

Метки обеспечивают возможность идентификации операторов для последующей ссылки на них из операторов `break` и `continue` по имени метки. Правила определения имён для меток аналогичны правилам именования переменных и констант. Синтаксис определения метки имеет следующий вид:

имя_метки : оператор

Оператор **`continue`** используется в сочетании с операторами цикла. Этот оператор позволяет прервать исполнение операторов тела цикла и перейти к выполнению следующей итерации этого оператора цикла, пропустив все следующие за ним операторы тела цикла.

В следующем примере на экран выводятся только нечетные числа:

Листинг 3.6.10.

```
<html><head>
<title>Пример 3.6.10. Оператор continue</title>
</head>
<body>
<h2>Пример 3.6.10. Использование оператора continue в цикле while</h2>
<h3>Вывод на экран нечётных чисел</h3>
<script type="text/javascript">
var x = 0;
while (x < 10)
  {
    x++;
    if (x % 2 == 0)
      {
        continue;
      }
```

```
    alert(x);
  }
</script></body></html>
```

В теле цикла `while` находятся три оператора. Вторым оператором является оператор `if`, в условии которого операция сравнения `(x % 2 == 0)` выдаёт `true`, если x чётно. В этом случае выполняется оператор `continue` и оператор `alert(x)` не выполняется. Если операция сравнения выдаёт `false`, то оператор `continue` не выполняется, `alert(x)` срабатывает и выдаёт на экран нечётное значение перменной `x`.

Оператор `continue` может использоваться и в сочетании с меткой. В этом случае нарушается предусмотренный по умолчанию порядок выполнения цикла, поскольку переход осуществляется к помеченному оператору. Пример использования метки показан в листинге 3.6.11.

Листинг 3.6.11.

```
<html><head>
<title>Пример 3.6.11. Оператор continue и метка в цикле for</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
<h2>Пример 3.6.11. Использование оператора continue и метки в цикле
for</h2>
<h3>Вывод на экран значений главной диагонали двумерной матрицы<br/>
случайных чисел a[i,j]=i*j*0.1*Math.random()</h3>
<script type="text/javascript">
var n=m=6;
// создаём массив-матрицу случайных чисел размером n*m
var a=new Array(n);
for (var i=0; i<n; i++) {
    a[i]=new Array(m)
    for (var j=0; j<m; j++)
     a[i][j]=i*j+0.1*Math.random();
}
// печатаем значения главной диагонали матрицы
m1: for (var i=0; i<n; i++)
      for (var j=0; j<m; j++)
          if (i==j) {
      document.write("<p>"+a[i][j].toFixed(3)+"</p>");
      continue m1
          }
          else
      continue;
</script></body></html>
```

Из примера можно понять, что, если во вложенном цикле после `continue` указывается метка, идентифицирующая внешний цикл, то итерации вложенного цикла прекращаются, когда выполняется переход на новую итерацию помеченного внешнего цикла.

Оператор break (прервать) также, как и `continue` используется в сочетании с операторами организации цикла. Однако, если оператор `continue` приводит к переходу к следующему циклу, то на операторе `break` циклы вообще прекращаются. Оператор `break` используется также с оператором `switch`, который рассматривается далее. В следующем примере оператор `break`, также как и оператор `continue` применяется в сочетании с оператором `if`.

Листинг 3.6.12

```
<html><head>
<title>Пример 3.6.12. Оператор break в цикле while</title>
</head>
<body>
<h2>Пример 3.6.12. Использование оператора break в цикле while</h2>
<h3>Вывод на экран последовательности целых чисел<br/>
от нуля до заданного числа</h3>
<script type="text/javascript">
var x = 0, br;
br=parseInt(prompt("Введите число для прерывания цикла от 1 до 40", ""));
document.write("<p>");
while (x < 40)
  {
  if (x > br)
    {
      break;
    }
  document.write(x+"  ");
  x++;
  }
document.write("</p>");
</script></body></html>
```

В этом примере вводится число, при котором следует прервать выполнение оператора `while`. Цикл оператора `while` будет выполняться до тех пор, пока значение `x` не станет больше значения переменной `br`, и тогда цикл прервется. Например, если вы введете в окне запроса значение 4, на печать будут выведены окна предупредительных сообщений со значениями 0 1 2 3 4.

Оператор `break` можно использовать совместно с меткой.

Оператор switch (перейти к…) позволяет выполнить один или несколько операторов, если значение указанного выражения совпадает с меткой. Синтаксис оператора имеет следующий вид:

```
switch (выражение)
{
case метка1;
группа операторов
break;
case метка2;
группа операторов
break;
. . .
default:
группа операторов
break;
}
```

В условии оператора `switch` может стоять имя переменной или выражение, возвращающее значение любого типа.

Тело оператора `switch` заключается в операторные скобки и состоит из повторяющихся ключевых слов `case`, после которых стоят метки. При срабатывании оператора `switch` проверяется совпадение метки и значения, полученного при вычислении выражения. При совпадении выполняется группа операторов, стоящая после метки.

Каждая группа операторов должна заканчиваться оператором `break`, в противном случае будут выполняться все последующие за ней операторы.

Если совпадения не обнаружены, выполняется группа операторов, стоящая после

метки **default**.

В листинге 3.6.13 приведён пример, в котором в окно запрос/ответ вводится целое число, а в программе с помощью оператора `switch` определяется совпадение введённого числа с меткой, которой помечен соответствующий оператор `alert()`, выводящий введённое число в текстовом виде.

Листинг 3.6.13.

```
<html><head>
<title>Пример 3.6.13. Оператор switch</title>
</head>
<body>
<h2>Пример 3.6.13. Использование оператора switch</h2>
<h3>Вывод на экран вводимого целого числа в текстовом виде</h3>
<script type="text/javascript">
var yourchoice;
yourchoice = parseInt(prompt("Загадайте целое число  от 1 до 4. Можно и
другие, но ...", "1, 2, 3 или 4"))
switch (yourchoice)
{
case 1:
  alert("Вы ввели число: один");
  break;
case 2:
  alert("Вы ввели число: два");
  break;
case 3:
  alert("Вы ввели число: три");
  break;
case 4:
  alert("Вы ввели число: четыре");
  break;
default: //эта метка предусмотрена для перехода по умолчанию
  alert("Вы не ввели число, попадающее в промежуток от 1 до 4");
  break;
}
</script></body></html>
```

3.6.5. Примеры типовых алгоритмов

Во всех примерах предварительно создаются массивы случайных чисел и по таким образом полученным данным демонстрируются некоторые элементарные алгоритмы вычисления числовых характеристик (статистических характеристик), часто используемых на практике. При обновлении документа массивы заполняются новыми значениями и, соответственно, меняются вычисляемые значения. Точные теоретические величины значений, получаемых по массиву случайных чисел, известны из теории вероятностей (вероятностные характеристики). Вероятностные характеристики можно использовать для проверки правильности расчёта статистических характеристик при отладке и опробовании программ.

Необходимо обратить внимание, что во всех приведённых ниже программах используются так называемые **динамические массивы**, которые в отличие от **статических массивов** создаются и увеличиваются в размерах по мере необходимости.

Пример **вычисления суммы и среднего арифметического** по значениям одномерного массива с использованием оператора цикла с постусловием (используется цикл `do...while`) приведён в листинге 3.6.14.

Можно выделить пять основных блоков программы (или этапов выполнения программы). Собственно вычисление суммы и среднего выполняется в четвёртом и пятом

блоках, а первые блоки являются вспомогательными. Опишем содержание каждого блока;

1) описываются необходимые переменные;

2) присваиваются необходимые значения некоторым переменным (исходные данные);

3) создаётся (генерируется) и выводится в документ массив случайных чисел. Значения массива при выводе округляются до трёх знаков после десятичной точки. Следует понимать, что в памяти компьютера эти числа не округлены.

4) созданный выше массив используется для накопления и вывода суммы;

5) полученная сумма используется для расчёта и вывода среднего арифметического.

Листинг 3.6.14.

```
<html><head>
<title>Пример 3.6.14. Сумма и среднее с оператором do...while</title>
</head>
<body>
<h2>Пример 3.6.14. Вычисление суммы и среднего арифметического</h2>
<h3>Используется оператор цикла с постусловием (do...while)</h3>
<script type="text/javascript">
// 1) описываем необходимые переменные
var n, i=sum=0, sr, a=new Array();
// 2) вводим исходные данные
n=20;
// 3)создаём и выводим  массив случайных чисел;
do {
    i=i+1;
    a[i]=Math.random()
}
while (i<n);
i=0;
document.write("<p>Массив случайных чисел:</p><p>")
do {
    i=i+1;
    document.write(a[i].toFixed(3)+"  ");
}
while (i<n); document.write("</p>");
// 4) накапливаем и выводим сумму
i=0;
do {
    i=i+1;
    sum=sum+a[i];
}
while (i<n);
document.write("<p>Сумма = "+sum.toFixed(3)+"</p>");
// 5) вычисляем и выводим среднее арифметическое
sr=sum/a.length;
document.write("<p>Среднее арифметическое = "+sr.toFixed(3)+"</p>");
</script></body></html>
```

В листинге 3.6.15 приводится пример определения экстремумов (минимального и максимального значений) по значениям одномерного массива. В примере используется оператор цикла с предъусловием (цикл `while`). Этапы выполнения программы поясняются в комментариях.

Листинг 3.6.15.

```
<html><head>
<title>Пример 3.6.15. Экстремумы с оператором while</title>
</head> <body>
```

```
<h2>Пример 3.6.15. Вычисление экстремумов (минимального и максимального
значений</h2>
<h3>Используется оператор цикла с предусловием (while)</h3>
<script type="text/javascript">
// 1) описываем переменные
var n, i, min, max, a=new Array();
// 2) вводим исходные данные
n=20;
// 2)генерируем и выводим массив случайных чисел
i=0
var a=new Array();
while (i<n) {
    i=i+1;
    a[i]=Math.random()
}
//
i=0;
document.write("<p>Массив случайных чисел:</p><p>")
while (i<n) {
    i=i+1;
    document.write(a[i].toFixed(3)+"  ");
}
document.write("</p>");
// 3)вычисляем и выводим минимум
i=0;
min=a[1];
while (i<n) {
    i=i+1;
    if (a[i]<min) min=a[i];
}
document.write("<p>Минимум = "+min.toFixed(3)+"</p>");
// 4)вычисляем и выводим максимум
i=0;
max=a[1];
while (i<n) {
    i=i+1;
    if (a[i]>max) max=a[i];
}
document.write("<p>Максимум = "+max.toFixed(3)+"</p>");
</script></body></html>
```

Вычисление суммы и среднего арифметического по значениям двумерного массива показано в листинге 3.6.16. В примере используется цикл с переменной-параметром (цикл for).

Листинг 3.6.16.

```
<html><head>
<title>Пример 3.6.16. Сумма и среднее по значениям двумерного
массива</title>
</head>
<body>
<h2>Пример 3.6.16. Вычисление суммы и среднего арифметического по
значениям двумерного массива.</h2>
<h3>Используется оператор цикла с переменной параметром (for)</h3>
<script type="text/javascript">
// 1) описываем переменные
var n, i, j, sum, sr, a=new Array();
// 2) вводим исходные данные
n=6;
```

```
// 3) создаём пустой двумерный массив
a=new Array(n);
    for(j=1; j<=n; j++)
       a[j]=new Array(n)
// 4) заполняем массив случайными числами:
for (i=1; i<=n; i++){
    for(j=1; j<=n; j++)
       a[i][j]=Math.random();
}
// 5) выводим массив в виде таблицы
document.write("<p>Матрица случайных чисел:</p>");
document.write("<table>");
for (i=1; i<=n; i++) {
    document.write("<tr>");
    for(j=1; j<=n; j++) {
       document.write("<td>"+a[i][j].toFixed(3)+" </td>");
    }
    document.write("</tr>");
}
document.write("<table>");
// 6) рассчитываем и выводим сумму значений массива
sum=0;
for (i=1; i<=n; i++)
    for(j=1; j<=n; j++)
       sum=sum+a[i][j];
document.write("<p>Сумма = "+sum.toFixed(3)+"</p>");
// 7) рассчитываем и выводим среднее арифметическое
sr=sum/(n*n);
document.write("<p>Среднее арифметическое = "+sr.toFixed(3)+"</p>");
</script></body></html>
```

Вычисление сумм и средних арифметических по значениям каждого столбца двумерного массива показано в листинге 3.6.17. В примере используется цикл `for`.

Листинг 3.6.17.

```
<html><head>
<title>Пример 3.6.17. Суммы и средние по столбцам двумерного
массива</title>
</head>
<body>
<h2>Пример 3.6.17. Вычисление сумм и средних арифметических по столбцам
двумерного массива.</h2>
<h3>Используется оператор цикла с переменной-параметром (for)</h3>
<script type="text/javascript">
// 1) описываем переменные
var n, i, j, sr, a=new Array(), sum=new Array(), sr=new Array();
// 2) вводим исходные данные
n=6;
// 3) создаём пустой двумерный массив
a=new Array(n);
    for(j=1; j<=n; j++)
       a[j]=new Array(n)
// 4) заполняем двумерный массив случайными числами,
// причём числа в столбцах вычисляются по формуле:
// Math.random+(j-1), j=1,2,3; j-индекс столбца
for (i=1; i<=n; i++){
    for(j=1; j<=n; j++)
       a[i][j]=Math.random()+(j-1);
}
```

```
// 5) выводим двумерный массив в виде таблицы
document.write("<p>Матрица случайных чисел:</p>");
document.write("<table>");
for (i=1; i<=n; i++) {
    document.write("<tr>");
    for(j=1; j<=n; j++) {
       document.write("<td>"+a[i][j].toFixed(3)+" </td>");
    }
    document.write("</tr>");
}
document.write("<table>");
// 6) рассчитываем суммы по столбцам и запоминаем их
// в одномерном массиве
for (j=1; j<=n; j++) {
    sum[j]=0;
    for(i=1; i<=n; i++) {
       sum[j]=sum[j]+a[i][j];
    }
}
// 7) выводим суммы
document.write("<p>Суммы по столбцам:</p>");
for (j=1; j<=n; j++)
    document.write(sum[j].toFixed(3)+" ");
// 8) рассчитываем, запоминаем и выводим средние арифметические
document.write("<p>Средние арифметические по столбцам:</p>");
for (j=1; j<=n; j++) {
    sr[j]=sum[j]/n;
    document.write(sr[j].toFixed(3)+" ");
}
</script></body></html>
```

Следующий пример отличается от примера в листинге 3.6.16 тем, для вычисления среднего арифметического не используются значения, выходящие за установленные пределы. Так, например, строятся фильтры, отсеивающие ошибочные значения. В примере используется цикл `for` совместно с операторами `if`, `continue` и меткой.

Листинг 3.6.18.

```
<html><head>
<title>Пример 3.6.18. Среднее по отфильтрованным значениям двумерного
массива</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
<h2>Пример 3.6.18. Вычисление суммы и среднего арифметического по
значениям двумерного массива.</h2>
<h3>Используется оператор цикла с переменной параметром (for)</h3>
<script type="text/javascript">
// 1) описываем переменные
var n, i, j, sum, sr, k, min, max, a=new Array();
// 2) вводим исходные данные
n=6; min=-0.4; max=0.4;
// 3) создаём пустой двумерный массив
a=new Array(n);
    for(j=1; j<=n; j++)
       a[j]=new Array(n)
// 4) заполняем массив случайными числами:
for (i=1; i<=n; i++){
    for(j=1; j<=n; j++)
```

```
       a[i][j]=Math.random()-0.5;
}
// 5) выводим массив в виде таблицы
document.write("<p>Матрица случайных чисел:</p>");
document.write("<table>");
for (i=1; i<=n; i++) {
    document.write("<tr>");
    for(j=1; j<=n; j++) {
       document.write("<td>"+a[i][j].toFixed(3)+" </td>");
    }
    document.write("</tr>");
}
document.write("<table>");
// 6) рассчитываем сумму значений массива
sum=0;
for (i=1; i<=n; i++)
m1: for(j=1; j<=n; j++)
       if (a[i][j]>max||a[i][j]<min) {
           document.write("<p>Значение "+a[i][j].toFixed(3)+"(строка
"+i+"; столбец "+j+") исключено из расчётов.</p>");
           continue m1
       }
       else sum=sum+a[i][j];
// 7) рассчитываем и выводим среднее арифметическое
sr=sum/(n*n);
document.write("<p>Среднее арифметическое = "+sr.toFixed(3)+"</p>");
</script></body></html>
```

3.6.6. Упражнения

Веб-страницы с выполненными упражнениями необходимо оформлять на своём персональном сайте так же, как это вы делали в предыдущих упражнениях. Не забывайте присоединять стили, созданные при освоении раздела, посвящённого HTML и CSS, чтобы сохранялось выбранное стилевое оформление всех страниц сайта. Помните также о тэге <meta ...> для указания кодировки UTF-8.

Примеры некоторых упражнений можно посмотреть у студента Петрова на сайте pvn.ho.ua.

Упражнение 3.6.1.

Создать html-документ с программой на языке Javascript, изменив пример из листинга 3.6.1. При этом необходимо учесть следующие условия:

1) повторите пример из листинга 3.6.1, заменив в нём первый полный оператор `if` двумя сокращёнными операторами `if`, а два последних сокращённых оператора `if` одним полным оператором `if`;

2) замените все выводы результатов методом `alert()` на вывод результатов методом `document.write()`;

3) в этом же html-документе поместите ответы на первые три контрольных вопроса из пункта 3.6.7 (с 1-го по 3-й);

Код упражнения необходимо сохранить в файле exercise_3_6_1.html и оформить его отображение на своём сайте так, как это вы делали с предыдущими упражнениями.

Напоминание! Полный оператор `if` содержит ключевые слова `if` и `else`, а сокращённый — только лишь `if` (без `else`).

Упражнение 3.6.2.

Напишите html-документ с программой на языке Javascript, в котором:

1) с помощью метода prompt() вводится целое число;

2) с помощью одного условного оператора делается проверка условия на чётность/нечётность введённого числа и, в зависимости от выполнения условия с помощью метода document.write() выводится введённое число и текстовое сообщение о чётности/нечётности числа.

3) в этом же html-документе поместите ответы на два контрольных вопроса из пункта 3.6.7 (4-й и 5-й вопросы);

Код упражнения необходимо сохранить в файле exercise_3_6_2.html и оформить его отображение на своём сайте так, как это вы делали с предыдущими упражнениями.

Упражнение 3.6.3.

Напишите html-документ с программой на языке Javascript, в котором повторите пример из листинга 3.6.3, изменив его так, чтобы:

1) число циклов n вводилось с помощью метода prompt();

2) вывод переменной цикла осуществлялся с помощью метода document.write();

3) выводились бы только лишь чётные значения переменной цикла;

4) в этом же html-документе поместите ответы на четыре контрольных вопроса из пункта 3.6.7 (с 6-го по 9-й).

Код упражнения необходимо сохранить в файле exercise_3_6_3.html и оформить его отображение на своём сайте так, как это вы делали с предыдущими упражнениями.

Упражнение 3.6.4.

Напишите html-документ с программой на языке Javascript, в котором повторите пример из листинга 3.6.4, изменив его так, чтобы:

1) число циклов n вводилось с помощью метода prompt();

2) вывод переменной цикла осуществлялся с помощью метода document.write();

3) выводились бы только лишь нечётные значения переменной цикла;

4) в этом же html-документе поместите ответы на три контрольных вопроса из пункта 3.6.7 (с 10-го по 12-й).

Код упражнения необходимо сохранить в файле exercise_3_6_4.html и оформить его отображение на своём сайте так, как это вы делали с предыдущими упражнениями.

Упражнение 3.6.5.

Напишите html-документ с программой на языке Javascript, в котором повторите пример из листинга 3.6.5, изменив его так, чтобы:

1) число циклов n вводилось с помощью метода prompt(), при этом: $n > 10$;

2) вывод переменной цикла осуществлялся с помощью метода document.write();

3) выводились бы значения переменной-параметра цикла только лишь попадающие в интервал [n-2, n-10];

4) в этом же html-документе поместите ответы на два контрольных вопроса из пункта 3.6.7 (13-й и 14-й).

Код упражнения необходимо сохранить в файле exercise_3_6_5.html и оформить его отображение на своём сайте так, как это вы делали с предыдущими упражнениями.

Упражнение 3.6.6.

Напишите html-документ с программой на языке Javascript, в котором повторите пример из листинга 3.6.6, изменив его так, чтобы:

1) в документ выводилось лишь два параграфа, в одном из них выбирался бы день недели «Вторник», а во втором - «Пятница» с добавлением текста « - я должен заниматься программированием»;

2) поиск в массиве дней недели необходимого дня осуществлялся бы с помощью оператора цикла for с вложенным оператором if, а в качестве условия для поиска вторника использовалась бы строка «Вторник», для поиска пятницы — индекс ;

3) в этом же html-документе поместите ответ на два контрольных вопроса из пункта 3.6.7 (15-й и 16-й).

Код упражнения необходимо сохранить в файле exercise_3_6_6.html и оформить его отображение на своём сайте так, как это вы делали с предыдущими упражнениями.

Упражнение 3.6.7.

Напишите html-документ с программой на языке Javascript, в котором повторите пример из листинга 3.6.14, изменив его так, чтобы:

1) число циклов n вводилось с помощью метода prompt();

2) вместо оператора do...while использовался бы цикл for;

3) в этом же html-документе поместите ответы на два контрольных вопроса из пункта 3.6.7 (17-й и 18-й).

Код упражнения необходимо сохранить в файле exercise_3_6_7.html и оформить его отображение на своём сайте так, как это вы делали с предыдущими упражнениями.

Упражнение 3.6.8.

Напишите html-документ с программой на языке Javascript, в котором повторите пример из листинга 3.6.15, изменив его так, чтобы:

1) число циклов n вводилось с помощью метода prompt();

2) вместо оператора while использовался бы цикл for;

3) в этом же html-документе поместите ответы на четыре контрольных вопроса из пункта 3.6.7 (с 19-го по 22-й).

Код упражнения необходимо сохранить в файле exercise_3_6_8.html и оформить его отображение на своём сайте так, как это вы делали с предыдущими упражнениями.

Упражнение 3.6.9.

Напишите html-документ с программой на языке Javascript, в котором повторите пример из листинга 3.6.16, изменив его так, чтобы:

1) число циклов n вводилось с помощью метода prompt();

2) для вычисления суммы и среднего арифметического не использовались бы значения последнего столбца двумерной матрицы;

3) в этом же html-документе поместите ответы на два контрольных вопроса из пункта 3.6.7 (23-й и 24-й).

Код упражнения необходимо сохранить в файле exercise_3_6_9.html и оформить его отображение на своём сайте так, как это вы делали с предыдущими упражнениями.

Упражнение 3.6.10.

Напишите html-документ с программой на языке Javascript, в котором повторите пример из листинга 3.6.17, изменив его так, чтобы:

1) число циклов n вводилось с помощью метода prompt(), причём n>2 и при вводе n <3

выводилось бы сообщение с просьбой повторить ввод;

2) для вычисления сумм и средних арифметических не использовались бы первое и последнее значения в каждом столбце двумерной матрицы;

3) в этом же html-документе поместите ответы на два контрольных вопроса из пункта 3.6.7 (25-й и 26-й).

Код упражнения необходимо сохранить в файле exercise_3_6_10.html и оформить его отображение на своём сайте так, как это вы делали с предыдущими упражнениями.

Упражнение 3.6.11.

Напишите html-документ с программой на языке Javascript, в котором повторите пример из листинга 3.6.18, изменив его так, чтобы:

1) число циклов n вводилось с помощью метода prompt(), причём n>2 и при вводе n <3 выводилось бы сообщение с просьбой повторить ввод;

2) для вычисления суммы и среднего арифметического значений главной диагонали матрицы использовался бы оператор if без использования метки и оператора continue;

3) в этом же html-документе поместите ответы на два контрольных вопроса из пункта 3.6.7 (27-й и 28-й).

Код упражнения необходимо сохранить в файле exercise_3_6_11.html и оформить его отображение на своём сайте так, как вы это делали с предыдущими упражнениями.

Упражнение 3.6.12.

Создать html-документ с программой на языке Javascript для генерации и отображения в документе двумерной квадратной матрицы случайных вещественных чисел. При этом необходимо учесть следующие условия:

1) входные данные: ***n, a, b, l*** - вводятся с помощью метода prompt();

2) размер матрицы: ***n*n,*** **где** ***n*** - целое число, ***2< n <10***;

3) случайные числа задаются с помощью метода Math.random() на интервале ***[a,b]***, где ***a*** и ***b*** — вещественные числа, ***a<b***;

4) при создании и отображении матрицы необходимо использовать оператор цикла for (оператор цикла с переменной-параметром);

5) вывод матрицы в документ реализовать в виде html-таблицы с использованием метода document.wrtite();

6) значения элементов матрицы должны быть округлены с точностью до ***l*** знаков после десятичной точки.

Код упражнения необходимо сохранить в файле exercise_3_6_12.html и оформить его отображение на своём сайте так, как вы это делали с предыдущими упражнениями.

Упражнение 3.6.13.

Создать html-документ с программой на языке Javascript, в которой необходимо повторить программу из упражнения 3.6.12 со следующим дополнением:

1) рассчитать средние арифметические значения по данным каждого столбца матрицы;

2) средние арифметические значения вывести в виде строки под ранее выведенной матрицей исходных значений с округлением до ***l*** знаков после десятичной точки;

Код упражнения необходимо сохранить в файле exercise_3_6_13.html и оформить его отображение на своём сайте так, как это вы делали с предыдущими упражнениями.

3.6.7. Контрольные вопросы

1. Синтаксис полного оператора `if`?
2. Синтаксис сокращённого оператора `if`?
3. В каком случае можно не использовать операторные скобки в операторе `if`?
4. Синтаксис условного оператора?
5. В каких случаях можно использовать условный оператор, вместо оператора `if`?
6. Операторы цикла ?
7. Ключевые слова оператора цикла с предусловием?
8. Выполнится ли оператор цикла с предусловием, если условие в «шапке» оператора цикла с самого начала ложно?
9. Будет ли ошибкой, если в теле оператора цикла с предусловием не будет ни одного оператора?
10. Ключевые слова оператора цикла с постусловием?
11. Выполнится ли оператор цикла с постусловием, если условие в «шапке» оператора цикла с самого начала ложно?
12. Будет ли ошибкой, если в теле оператора цикла с постусловием не будет ни одного оператора?
13. Чему равна переменная-параметр в цикле for после завершения оператора цикла?
14. Будет ли ошибкой, если в теле оператора цикла for не будет ни одного оператора?
15. Можно ли использовать оператор цикла for для работы с ассоциативным массивом (массивом, индексы которого — неупорядоченная последовательность чисел или текстовых значений)?
16. Пример вложенного цикла for?
17. Пример цикла for...in?
18. Можно ли использовать оператор цикла for...in для работы с обычным массивом (массивом, индексы которого — упорядоченная последовательность целых чисел)?
19. Назначение оператора continue?
20. Назначение оператора break?
21. Может ли использоваться метка совместно с оператором break?
22. Ключевые слова оператора switch?
23. Пример операторов для вычисления суммы по значениям одномерного массива?
24. Пример операторов для поиска минимума среди значений одномерного массива?
25. Пример операторов для поиска максимума среди значений одномерного массива?
26. Пример операторов для вычисления суммы по значениям двумерного массива?
27. Пример операторов для вычисления суммы по значениям главной диагонали двумерной матрицы?
28. Пример операторов для поиска минимума среди значений главной диагонали двумерной матрицы?
29. Пример операторов для вычисления суммы по значениям под главной диагональю двумерной матрицы?
30. Пример операторов для поиска максимума среди значений над главной диагональю двумерной матрицы?

3.5.8. Источники

1. Справочник по языку Javascript - http://javascript.ru/manual