Федеральное государственное бюджетное образовательное учреждение высшего образования «Кемеровский государственный университет» Новокузнецкий институт (филиал)

Факультет информатики, математики и экономики

Кафедра информатики и общетехнических дисциплин

Г. Н. Бойченко

Программирование на JavaScript

Методические указания к выполнению лабораторных работ для обучающихся по направлению подготовки

44.03.05 Педагогическое образование (с двумя профилями подготовки)

направленности (профили) подготовки «Математика и Информатика», «Информатика и Физика», «Технология и Информатика»

Новокузнецк
2020

УДК 004.421 (075.8)
ББК 32.973.434

Бойченко Г. Н.

Б779 Программирование на JavaScript: метод. указ. к выполнению лабораторных работ для обучающихся по направлению подготовки 44.03.05 Педагогическое образование (с двумя профилями подготовки), направленности (профили) подготовки «Математика и Информатика», «Информатика и Физика», «Технология и Информатика» / Г. Н. Бойченко; Новокузнец. ин-т (фил.) Кемеров. гос. ун-та. – Новокузнецк : НФИ КемГУ, 2020. – 56 с.

Приводятся методические указания к выполнению 6 лабораторных работ по дисциплине «Программирование на JavaScript», раздел «Основы языка JavaScript». В описании лабораторных работ представлены: тема, цель работы, теоретический материал по теме, задания к выполнению.

Методические указания предназначены для студентов очной и заочной форм обучения, обучающихся по направлению подготовки 44.03.05 Педагогическое образование (с двумя профилями подготовки), направленности (профили) подготовки «Математика и Информатика», «Информатика и Физика», «Технология и Информатика».

Рекомендовано
на заседании кафедры информатики
и общетехнических дисциплин
21 октября 2020 года, протокол №3

Заведующий кафедрой ИОТД

_____ И.В. Сликишина

Утверждено
методической комиссией факультета
информатики, математики и экономики
12 ноября 2020 года, протокол №4.

Председатель методкомиссии ФИМЭ

_____ Г.Н. Бойченко

Текст представлен в авторской редакции

Содержание

Введение

Язык программирования Javascript является динамическим скриптовым языком программирования высокого уровня. Он отличается мультипарадигменностью, поддерживая функциональный, императивный, событийно-ориентированный стили программирования. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

JavaScript обладает также пропедевтической ценностью, позволяя сочетать при обучении информатике интенсивную практику программирования и широту используемых технологий. Преподавание данного языка в школе позволяет создать базу для изучения веб-программирования, использовать на уроках творческие проекты. Соответствующий курс позволяет обеспечить углубленный уровень изучения информатики и его имеет смысл включать в элективные курсы углубленного уровня подготовки. JavaScript — также подходящий язык для обучения программированию игр. По сравнению с альтернативами, он функционально достаточен, прост в изучении и в применении, снижает сложность для обучения, мотивирует обучаемых делиться своими играми с другими.

В результате освоения дисциплины «Программирование на JavaScript» будущие учителя информатики должны:

знать:

- терминологию, используемую в клиент-серверных и веб-приложениях;
- синтаксис языка JavaScript;
- встроенные функции языка JavaScript;
- модель взаимодействия скриптов и содержимого веб-страниц;
- принципы объектно-ориентированного дизайна и проектирования с использованием JavaScript.;

уметь:

- создавать клиентские скрипты на языке JavaScript;
- тестировать и отлаживать клиентские скрипы.

владеть:

- инструментальными средствами написания, отладки и тестирования JavaScript-кода.

Предлагаемые методические указания адресованы студентам очной и заочной форм обучения, обучающихся по направлению подготовки 44.03.05 Педагогическое образование (с двумя профилями подготовки), направленность (профиль) подготовки «Математика и Информатика», «Информатика и Физика», «Технология и Информатика».

Цель настоящих методических указаний – дать студентам возможность углубить и закрепить полученные на лекциях знания и умения, а также приобрести опыт применения этих знаний на практике при самостоятельном решении различных практических задач.

Методические указания содержат 6 лабораторных работ по дисциплине «Программирование на JavaScript», раздел «Основы языка JavaScript». В описании лабораторных работ представлены: тема, цель работы, теоретический материал по теме, задания к выполнению.

Лабораторная работа 1. Линейные алгоритмы. Ветвление.

Цель работы: научиться разрабатывать скрипты на языке JavaScript, реализующие линейные и разветвленные алгоритмы обработки данных.

Теоретический материал по теме

Переменные и литералы

Переменная – это контейнер, у которого есть имя. Переменные используются для хранения информации, которая может изменяться во время работы программы.

Литерал – это значение, указанное непосредственно в тексте программы.

JavaScript поддерживает следующие **типы переменных**:

Числовой тип данных (number) предназначен для обработки любых числовых значений, как целых, так и вещественных. Примеры чисел:

```
0
3
10000000
3.14
2345.789
.333333333333333333
6.02e23 // 6.02 X 10^23
1.4738223E32 // 1.4738223 X 10^32
```

Для работы с числами в JavaScript-программах используются поддерживаемые языком арифметические операторы, к которым относятся операторы сложения (+), вычитания (-), умножения (*) и деления (/).

Строковый тип данных (string) предназначен для обработки строк, т.е. последовательностей символов, заключенных в одинарные или двойные кавычки. Примеры строк:

```
"" // Это пустая строка: в ней ноль символов
'testing'
"3.14"
'name="myform"'
"Пи - это отношение длины окружности круга к его диаметру"
```

Логический тип данных (boolean) может принимать только два значения: true (истина) и false (ложь).

Переменные создаются и им присваиваются значения следующим образом:

```
let name = "Иван"          // string
let surname = "Иванов"     // string
let age = 20               // number
let married = false        // boolean
```

Обратите внимание, что:

- Когда создается (или объявляется) новая переменная, ее имени должно предшествовать слово `let`.
- Тип переменной определяется способом ее объявления:
  - если он заключен в кавычки, это строка;
  - если установлено значение true или false (без кавычек), это логическое значение;
  - если это число (без кавычек), оно числовое.
- Знак равенства называется оператором присваивания, так как он используется для присвоения значений переменным.
- Имена переменных должны начинаться с буквы или символа подчеркивания.

- Имена переменных не должны содержать пробелов.
- JavaScript чувствителен к регистру.
- Зарезервированные слова (т. е. слова, обозначающие действие или операцию в JavaScript) не могут использоваться в качестве имен переменных.

Выражения и операторы

Выражение – это фраза языка JavaScript, которая может быть вычислена интерпретатором для получения значения.

Операторы используются для выполнения операций с переменными и / или литералами и вычисления результата.

Арифметические операторы своими операндами принимают числовые значения (литералы или переменные) и возвращают одно числовое значение: + (оператор сложения), - (оператор вычитания), * (оператор умножения), / (оператор деления), %, (оператор взятия остатка от деления), ** (оператор возведения в степень).

Операторы инкремента и декремента: A++ (оператор постфиксного инкремента), A-- (оператор постфиксного декремента), ++A (оператор префиксного инкремента), --A (оператор префиксного декремента).

Операторы присваивания: = (оператор присваивания), *= (оператор присваивания с умножением), /= (оператор присваивания с делением), %= (оператор присваивания с взятием остатка от деления), += (оператор присваивания со сложением), -= (оператор присваивания с вычитанием).

Операторы сравнения: == (оператор проверки на равенство), != (оператор проверки на неравенство), === (оператор проверки на идентичность), !== (оператор проверки на неидентичность), > (оператор больше), >= (оператор больше или равно), < (оператор меньше), <= (оператор меньше или равно).

Логические операторы: && (оператор логического И), || (оператор логического ИЛИ), ! (оператор логического НЕ).

Объект Math является встроенным объектом, хранящим в своих свойствах и методах различные математические константы и функции:

Свойство объекта `Math`	**Описание**
`Math.E`	Число Эйлера, основание натуральных логарифмов, приблизительно равное 2,718.
`Math.LN2`	Натуральный логарифм из 2, приблизительно равен 0,693.
`Math.LN10`	Натуральный логарифм из 10, приблизительно равен 2,303.
`Math.LOG2E`	Двоичный логарифм из E, приблизительно равен 1,443.
`Math.LOG10E`	Десятичный логарифм из E, приблизительно равен 0,434.
`Math.PI`	Отношение длины окружности круга к его диаметру, приблизительно равно 3,14159.
`Math.SQRT1_2`	Квадратный корень из 1/2; или, что тоже самое, 1, делённая на квадратный корень из 2, приблизительно равен 0,707.
`Math.SQRT2`	Квадратный корень из 2, приблизительно равен 1,414.

Метод объекта `Math`	**Описание**
Math.abs(x)	Возвращает абсолютное значение числа.
Math.acos(x)	Возвращает арккосинус числа.
Math.acosh(x)	Возвращает гиперболический арккосинус числа.
Math.asin(x)	Возвращает арксинус числа.
Math.atan(x)	Возвращает арктангенс числа.

Метод объекта `Math`	Описание
Math.atanh(x)	Возвращает гиперболический арктангенс числа.
Math.atan2(y, x)	Возвращает арктангенс от частного своих аргументов.
Math.cbrt(x)	Возвращает кубический корень числа.
Math.ceil(x)	Возвращает значение числа, округлённое к большему целому.
Math.clz32(x)	Возвращает количество ведущих нулей 32-битного целого числа.
Math.cos(x)	Возвращает косинус числа.
Math.cosh(x)	Возвращает гиперболический косинус числа.
Math.exp(x)	Возвращает E^x, где x — аргумент, а E — число Эйлера (2,718…), основание натурального логарифма.
Math.expm1(x)	Возвращает exp(x), из которого вычли единицу.
Math.floor(x)	Возвращает значение числа, округлённое к меньшему целому.
Math.fround(x)	Возвращает ближайшее число с плавающей запятой одинарной точности, представляющее это число.
Math.hypot([x[, y[, …]]])	Возвращает квадратный корень из суммы квадратов своих аргументов.
Math.imul(x)	Возвращает результат умножения 32-битных целых чисел.
Math.log(x)	Возвращает натуральный логарифм числа ($\log_e$, также известен как ln).
Math.log1p(x)	Возвращает натуральный логарифм числа 1 + x ($\log_e$, также известен как ln).
Math.log10(x)	Возвращает десятичный логарифм числа.
Math.log2(x)	Возвращает двоичный логарифм числа.
Math.max([x[, y[, …]]])	Возвращает наибольшее число из своих аргументов.
Math.min([x[, y[, …]]])	Возвращает наименьшее число из своих аргументов.
Math.pow(x, y)	Возвращает основание в степени экспоненты, то есть, значение выражения x^y.
Math.random()	Возвращает псевдослучайное число в диапазоне от 0 до 1.
Math.round(x)	Возвращает значение числа, округлённое до ближайшего целого.
Math.sign(x)	Возвращает знак числа, указывающий, является ли число положительным, отрицательным или нулём.
Math.sin(x)	Возвращает синус числа.
Math.sinh(x)	Возвращает гиперболический синус числа.
Math.sqrt(x)	Возвращает квадратный корень числа.
Math.tan(x)	Возвращает тангенс числа.
Math.tanh(x)	Возвращает гиперболический тангенс числа.
Math.trunc(x)	Возвращает целую часть числа, убирая дробные цифры.

Инструкция if…else

Синтаксис:

```
if (условие)
   инструкция1
[else
   инструкция2]
```

`условие`	Выражение, которое является либо истинным, либо ложным.

инструкция1	Инструкция, выполняемая в случае, если значение "условие" истинно (true). Может быть любой инструкцией в том числе и вложенным if. Для группировки нескольких инструкций используется блок {...}. Когда никакого действия не требуется, может использоваться пустая инструкция.
инструкция2	Инструкция, выполняемая в случае, если значение "условие" ложно (false). Может быть любой инструкцией, в том числе и вложенным if. Инструкции тоже можно группировать в блок {...}.

Примеры:

```
if (username == null) username = "John Doe";
if ((address == null) || (address == "")) {
    address = "undefined";
    alert("Пожалуйста, укажите почтовый адрес.");
}
if (username != null)
   alert("Привет " + username + "\nДобро пожаловать на сайт.");
else {
   username = prompt("Добро пожаловать!\n Как вас зовут?");
   alert("Привет " + username);
}
```

Условный (тернарный) оператор часто используется в качестве укороченного варианта инструкции if…else.

Синтаксис:

```
condition ? ifTrue : ifFalse
```

Условный оператор принимает три операнда: условие, за которым следует знак вопроса (?), затем выражение, которое выполняется, если условие истинно, сопровождается двоеточием (:), и, наконец, выражение, которое выполняется, если условие ложно.

Примеры:

1)

```
    "The fee is " + (isMember ? "$2.00" : "$10.00")
```

2)

```
    var stop = false, age = 16;
    age > 18 ? location.assign("continue.html") : stop = true;
```

Диалоговые окна

Объект `window`, представляет открытое окно в браузере. Для отображения в браузере модальных диалоговых окон используются следующие методы объекта `window` - функции `alert(),confirm(),prompt()`.

Функция `alert()` предназначена для вывода в браузере предупреждающего модального диалогового окна с некоторым сообщением и кнопкой «ОК». При его появлении дальнейшее выполнение кода страницы прекращается до тех пор, пока пользователь не закроет это окно. Кроме этого, оно также блокирует возможность взаимодействия пользователя с остальной частью страницы.

Синтаксис:

```
alert(сообщение);
```

Пример:

```
  alert('Привет, мир!');
```

Функция `confirm()` отображает модальное диалоговое окно, которое содержит две кнопки (`OK` и `Cancel`), а так же опциональное (необязательное) текстовое сообщение.

Синтаксис:

```
result = window.confirm(message);
```

message	Опциональная (необязательная) строка, которая будет отображена в диалоговом окне
result	Булево значение, указывающее на нажатую кнопку. В переменную result возвращается: true - если пользователь нажал на кнопку «ОК»; false - в остальных случаях.

Примеры:

1)
```
let ok = confirm('Разрешаете ли вы использование cookie?');
alert(ok);
```
2)
```
if (window.confirm("Do you really want to leave?")) {
  window.open("exit.html", "Thanks for Visiting!");
}
```

Функция `prompt()` выводит модальное окно с заголовком `title`, полем для ввода текста, заполненным строкой по умолчанию `default` и кнопками `OK/CANCEL`.

Синтаксис:
```
result = prompt(title, default);
```

Вызов `prompt` возвращает то, что ввёл посетитель – строку или специальное значение `null`, если ввод отменён.

Примеры:

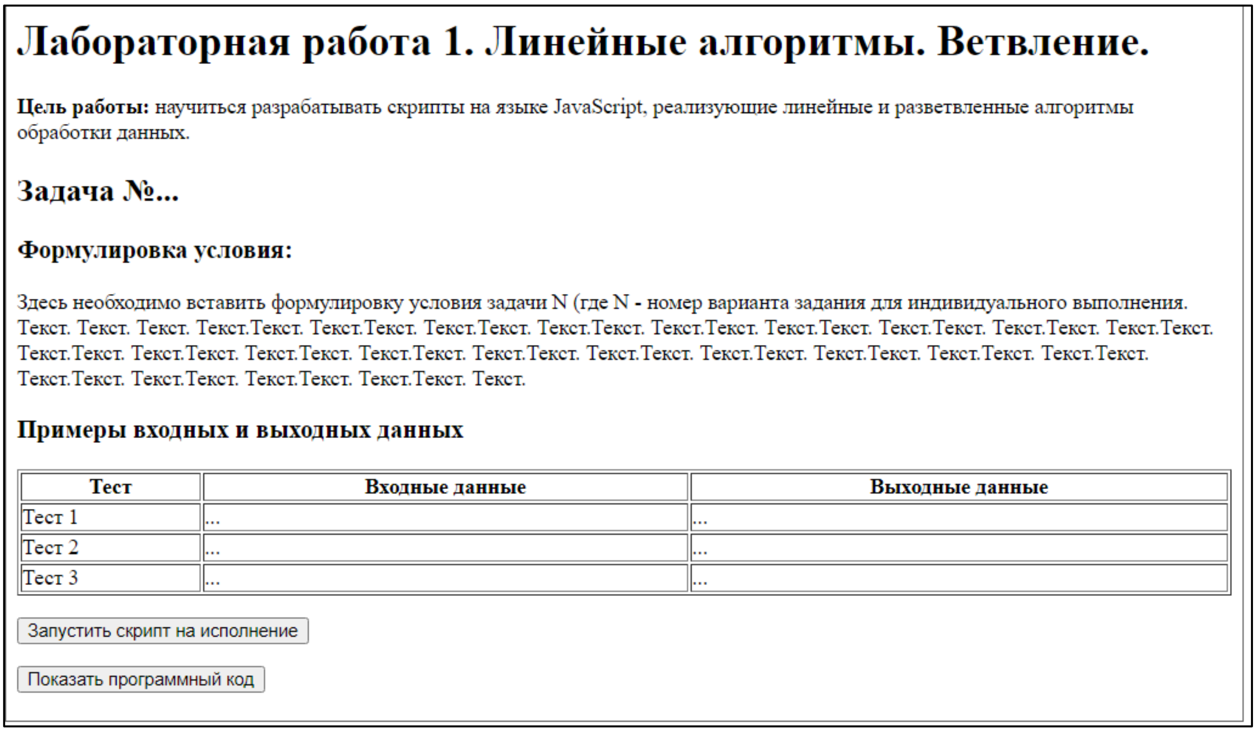
1)
```
let myName = prompt("Как тебя зовут?","");
alert("Привет " + myName + "!");
```
2)
```
let age = prompt('Сколько тебе лет?', 100);
alert('Тебе ${age} лет! '); // Тебе 100 лет!
```

Задания для выполнения

Задание 1. Разработайте скрипт, реализующий линейный алгоритм решения задачи. Ввод исходных данных осуществлять с использованием функции `promt`. Вывод полученных результатов осуществлять с использованием функции `alert`. Сохраните скрипт в файле **script_1_1.js**.

Задание 2. Разработайте скрипт, реализующий разветвленный алгоритм решения задачи. Ввод исходных данных осуществлять с использованием функции `promt`. Вывод полученных результатов осуществлять с использованием функции `alert`. Сохраните скрипт в файле **script_1_2.js**

Задание 3. Создайте по образцу html-документы **lab_1_1.html** и **lab_1_2.html** и подключите к ним внешние скрипты **script_1_1.js** и **script_1_2.js** соответственно. Для тестирования работоспособности каждого скрипта разработайте по 3 теста (включающих наборы входных и выходных данных). Выполните тестирование и отладку скриптов в браузере.



Лабораторная работа 1. Линейные алгоритмы. Ветвление.

Цель работы: научиться разрабатывать скрипты на языке JavaScript, реализующие линейные и разветвленные алгоритмы обработки данных.

Задача №...

Формулировка условия:

Здесь необходимо вставить формулировку условия задачи N (где N - номер варианта задания для индивидуального выполнения. Текст. Текст. Текст. Текст.

Примеры входных и выходных данных

Тест	Входные данные	Выходные данные
Тест 1	...	...
Тест 2	...	...
Тест 3	...	...

Запустить скрипт на исполнение

Показать программный код

Образец html-документа

Варианты индивидуальных задач к Заданию 1:

1. Даны катеты прямоугольного треугольника. Найти его гипотенузу и площадь.
2. Вычислить расстояние между двумя точками с координатами (x_1, y_1) и (x_2, y_2).
3. Найти длину окружности и площадь круга радиуса *R*.
4. Определить время падения камня на поверхность земли с высоты *H*.
5. Даны два действительных числа. Найти их сумму, разность и произведение.
6. Найти площадь поверхности и объем куба, если известно, что длина его ребра равна *a*.
7. Длина стороны равностороннего треугольника равна *a*. Найти его площадь и полупериметр.
8. Вычислить период *T* и частоту ν колебаний математического маятника длины *l*, если известно, что они связаны следующим соотношением: $T = \frac{1}{\nu} = 2\pi\sqrt{\frac{l}{g}}$, где $g = 9.8\frac{м}{c^2}$ — ускорение свободного падения.
9. Найти координаты точки (x, y), делящей отрезок *AB* в отношении $\frac{m}{n}$ (где *m* и *n* — натуральные числа), если известно, что его концы имеют координаты: $A(x_1, y_1)$, $B(x_2, y_2)$.
10. Вычислить высоты треугольника со сторонами *a, b* и *c*.
11. Вычислить площадь поверхности сферы и объем шара радиусом *R*.
12. Вычислить массу *l* м свинцовой трубы (плотность $11.3\frac{г}{см^3}$), толщина стенок которой *a* мм, а внутренний диаметр *r* мм.

13. Дан цилиндр единичного объема высотой *H*. Найти радиус его основания.
14. Вычислить площадь кольца, внутренний и внешний диаметры которого равны соответственно d_1 и d_2.
15. Вычислить процент материала, ушедшего в отходы, если из куба с ребром *a* был выточен шар радиусом *r* (*r*<*a*).
16. Найти площадь поверхности *S* конуса, заданного диаметром основания *D* и длиной образующей *L*.
17. Определить силу притяжения *F* между телами массы m_1 и m_2, находящимися на расстоянии *R* друг от друга ($F = G\frac{m_1 \cdot m_2}{R^2}$, где $G = 6.672 \cdot 10^{-11}\ Н \cdot м^2 / кг^2$ — гравитационная постоянная).
18. Даны действительные числа *x* и *y*. Вывести на экран значения выражений: $\frac{|x| - |y|}{1 + |x \cdot y|}$ и $\frac{|x| - |y|}{|x| + |y|}$.
19. Вычислить объем и площадь поверхности призмы, боковые грани которой — квадраты, а основанием служит равносторонний треугольник, вписанный в круг радиуса *R*.
20. Стороны треугольника равны *a, b* и *c*. Вычислить его углы.
21. Стальной вал, имеющий *l* мм длины и *d* мм в диаметре, обтачивается на токарном станке, причем его диаметр уменьшается при обточке на *s* мм. Вычислить, насколько уменьшается масса вала (плотность стали — $7.8\frac{г}{см^3}$).
22. Вычислить длину диагонали и площадь прямоугольника, вписанного в окружность радиуса *R*, если отношение его сторон равно *n*.
23. Вычислить площадь кольца, ширина которого равна *a*, а отношение внутреннего и внешнего радиусов кольца равно *b*.
24. Найти диаметр и площадь поперечного сечения трубы, пропускная способность которой позволяет заменить ею две трубы с диаметрами d_1 и d_2.
25. Найти объем и площадь боковой поверхности цилиндра, вписанного в правильную шестиугольную призму, у которой каждое ребро равно *a*.
26. Вычислить в равностороннем треугольнике сторону, высоту и площадь, если радиус вписанной в него окружности равен *r*.
27. Шар массой *M* с радиусом *R* и постоянной плотностью ρ притягивает материальную точку массой *m*, находящуюся на расстоянии *r* от центра шара ($r \ge R$). Найти силу притяжения по формуле $F = \frac{\gamma \cdot R^3}{r^2}$, где $\gamma = \frac{4\pi}{3} G\rho m$ ($G = 6.672 \cdot 10^{-11}\ Н \cdot м^2 / кг^2$ — гравитационная постоянная).
28. Треугольник задан тремя сторонами. Вычислить его медианы.

29. Дан четырехугольник со сторонами, длины которых равны *a, b, c* и *d*. Вычислить его периметр и площадь, если известно, что угол между сторонами *a* и *b* равен $\frac{\pi}{2}$.

30. Вычислить координаты *(x, y, z)* центра тяжести системы, состоящей из двух материальных точек с массами m_1 и m_2 и координатами (x_1, y_1, z_1) и (x_2, y_2, z_2). Указание: координаты центра тяжести *(x, y, z)* вычисляются по следующим формулам: $x = \frac{m_1 x_1 + m_2 x_2}{m_1 + m_2}$, $y = \frac{m_1 y_1 + m_2 y_2}{m_1 + m_2}$, $z = \frac{m_1 z_1 + m_2 z_2}{m_1 + m_2}$.

Варианты индивидуальных задач к Заданию 2:

1. Даны три действительных числа. Выбрать из них те, которые принадлежат интервалу (1, 3).
2. Числа *a* и *b* выражают длины катетов одного прямоугольного треугольника, а *c* и *d* — другого. Узнать, являются ли треугольники подобными.
3. Даны действительные числа *x*, *y* и *z*. Найти среди них наименьшее.
4. Клетка на шахматной доске определяется парой чисел. Определить, одного ли цвета две клетки.
5. Даны три действительных числа. Возвести в квадрат те из них, значения которых неотрицательны.
6. Упорядочить по возрастанию последовательность трех чисел *x, y, z* (вывести на печать заданные числа в порядке возрастания).
7. Даны действительные числа *x, y, z*. Вычислить *max*(*x* + *y* + *z*, *xyz*).
8. Если сумма трех попарно различных действительных чисел *x, y, z* меньше единицы, то наименьшее из этих чисел заменить полусуммой двух других; в противном случае заменить меньшее из *x* и *y* полусуммой двух оставшихся значений.
9. Каждое из чисел *a* и *b* отлично от нуля. Если они одного знака, то заменить меньшее из них большим; если же числа имеют разные знаки, присвоить каждому из них знак числа, меньшего по абсолютной величине.
10. Дано действительное число *X*. Напечатать в порядке убывания числа: X^2, *X*, $\sqrt{|X|}$.
11. Найти среди чисел *a, b* и *c* наименьшее и заменить им число, больше из них.
12. Даны действительные числа *a, b* и *c*. Утроить эти числа, если $a \le b \le c$, и заменить их абсолютными значениями, если это не так.
13. Даны действительные числа *a, b, c* и *d*. Если $a \le b \le c \le d$, то каждое число заменить наибольшим из них; если $a > b > c > d$, то числа оставить без изменения; в противном случае все числа заменяются их квадратами.
14. Вычислить, если это возможно, значение выражения: $\frac{x + y}{(x - 3) \cdot (y + 5)}$, где *x* и *y* — действительные числа.
15. Даны целые числа *m* и *n*. Если числа не равны, то заменить каждое из них одним и тем же числом, равным меньшему из исходных, а если равны, то заменить числа их квадратами.

16. Определить, какая из точек плоскости $A(x_1, y_1)$ и $B(x_2, y_2)$ находится ближе к началу координат.
17. Даны действительные числа *x, y*. Если *x* и *y* отрицательны, то каждое значение заменить его модулем; если отрицательно только одно из них, то оба значения увеличить на 0.5; если оба значения неотрицательны и ни одно из них не принадлежит отрезку [0.5, 2.0], то оба значения уменьшить в 10 раз; в остальных случаях *x* и *y* оставить без изменения.
18. Заменить каждое из чисел *a, b* и *c* значением $\frac{1}{2}\left[\max(a,b,c) + \min(a,b,c)\right]$.
19. Даны действительные положительные числа *x, y, z*. Выяснить, существует ли треугольник с длинами сторон *x, y, z*.
20. Найти квадрат наибольшего из двух чисел *a* и *b*.
21. Если значение переменной *w* не равно 0 и при этом котангенс от *w* меньше 0.5, тогда поменять знак у *w*, а если значение *w* равно 0, тогда присвоить *w* значение 1.
22. Даны два натуральных числа. Возвести в квадрат первое число, если оно больше второго, и оба эти числа, если это не так.
23. Даны натуральные числа *x* и *y* ($x \neq y$). Меньшее из этих чисел заменить их утроенным произведением, а большее — их полусуммой.
24. Значения переменных *A, B, C, D* вводятся с клавиатуры. Выяснить, есть ли среди них переменные, имеющие нулевые значения.
25. Даны действительные числа *a, b* и *c* ($a \neq 0$). Выяснить, имеет ли уравнение $ax^2 + bx + c = 0$ действительные корни.
26. Вычислить, если это возможно, значение выражения $\frac{1}{x+1} + \frac{1}{x-1}$, где *x* - действительное число.
27. Даны действительные числа *a, b* и *c*. Проверить, выполняются ли неравенства $a<b<c$.
28. На координатной плоскости построили квадрат, заданный координатами двух противоположных вершин, стороны которого параллельны осям координат. Определить, принадлежит ли точка с координатами *x, y* квадрату.
29. Даны два действительных числа. Заменить первое число нулем, если оно больше или равно второму. В противном случае утроить оба числа.
30. Две окружности заданы координатами своих центров и радиусами. Определить, имеют ли данные окружности общие точки.

Лабораторная работа 2. Циклические алгоритмы

Цель работы: научиться разрабатывать скрипты на языке JavaScript, реализующие циклические алгоритмы обработки данных с использованием циклов `for`, `while`, do…while.

Теоретический материал по теме

Цикл `for`

Цикл `for` позволяет выполнять определенную инструкцию фиксированное количество раз.

Синтаксис цикла `for`:

```
for(инициализация; условие; финальное выражение)
    инструкция
```

`Инициализация` - выражение (в том числе выражения присвоения) или определение переменных. Обычно используется, чтобы инициализировать счётчик. Это выражение может опционально объявлять новые переменные с помощью ключевого слова `var`. Эти переменные видимы не только в цикле, т.е. в той же области области видимости, что и цикл `for`. Результат этого выражения отбрасывается.

`Условие` - выражение, выполняющееся на каждой итерации цикла. Если выражение истинно, цикл выполняется. Условие не является обязательным. Если его нет, условие всегда считается истиной. Если выражение ложно, выполнение переходит к первому выражению, следующему за `for`.

`Финальное выражение` - выражение, выполняющееся в конце итерации цикла. Обычно используется для обновления или увеличения переменной счётчика.

`Инструкция` - инструкция, которое выполняется, когда условие цикла истинно (тело цикла). Чтобы выполнить в цикле множество `инструкций`, используется блок { ... } для их группировки.

Примеры:

1)
```
for(var count = 0; count < 10; count++)
    document.write(count + "<br>");
```
2)
```
for (var i = 0; i < 9; i++) {
   n += I;
}
```
3)
```
for(i = 0, j = 10; i < 10; i++, j)
    sum += i * j;
```
4)
```
for(;;) {           // бесконечный цикл
  ...
  if (какое-то условие) break;
}
```

Цикл `while`

Цикл **`while`** выполняет заданную инструкцию, пока истинно проверяемое условие.

Синтаксис цикла **`while:`**

```
while (условие)
  инструкция
```

Условие – логическое выражение, значение которого проверяется перед каждой итерацией цикла. Если значение истинно, то выполняется инструкция (тело цикла). Когда значение становится ложным, выполняется код, следующий за циклом `while`.

Инструкция - инструкция, которая исполняется каждый раз, пока истинно условие (тело цикла). Чтобы выполнить несколько инструкций в цикле, используется блочный оператор { ... } для их группировки.

Примеры:

1)
```
let count = 0;
while (count < 10) {
   document.write(count + "<br>");
   count++;
}
```
2)
```
n = 0;
x = 0;
while (n < 3) {
  n++;
  x += n;
  alert("n=" + n + ", x=" + x);
}
```
3)
```
let x = 500000;
alert("Начинается обратный отсчет...");
while (x > 0) {
   x--;
};
alert("Отбратный отсчет завершен!");
```

Цикл `do...while`

Цикл **`do...while`** выполняет указанное выражение до тех пор, пока условие не станет ложным.

Синтаксис цикла **`do...while:`**
```
do
   инструкция
while (условие);
```
Инструкция – инструкция (тело цикла), которая выполняется на каждой итерации цикла, пока условие истинно. Тело цикла выполняется как минимум один раз. Тело цикла может содержать несколько инструкций, которые необходимо сгруппировать в блок { ... }.

Условие - логическое выражение, значение которого вычисляется после каждой итерации цикла. Если значение истинно, то инструкция (тело цикла) выполняется еще раз. Когда значение становится ложным, выполняется код, следующий после цикла `do...while`.

Примеры:

1)
```
i = 0;
do {
   i += 1;
   document.write(i)
} while (i < 5);
```

2)

```
let passwordNotVerified = true;
do {
   let input = prompt("Введите пароль", "");
   if (input == password) {
      passwordNotVerified = false;
   }
   else {
      alert("Пароль неверный – повторите ввод")
   }
} while (passwordNotVerified == true)
```

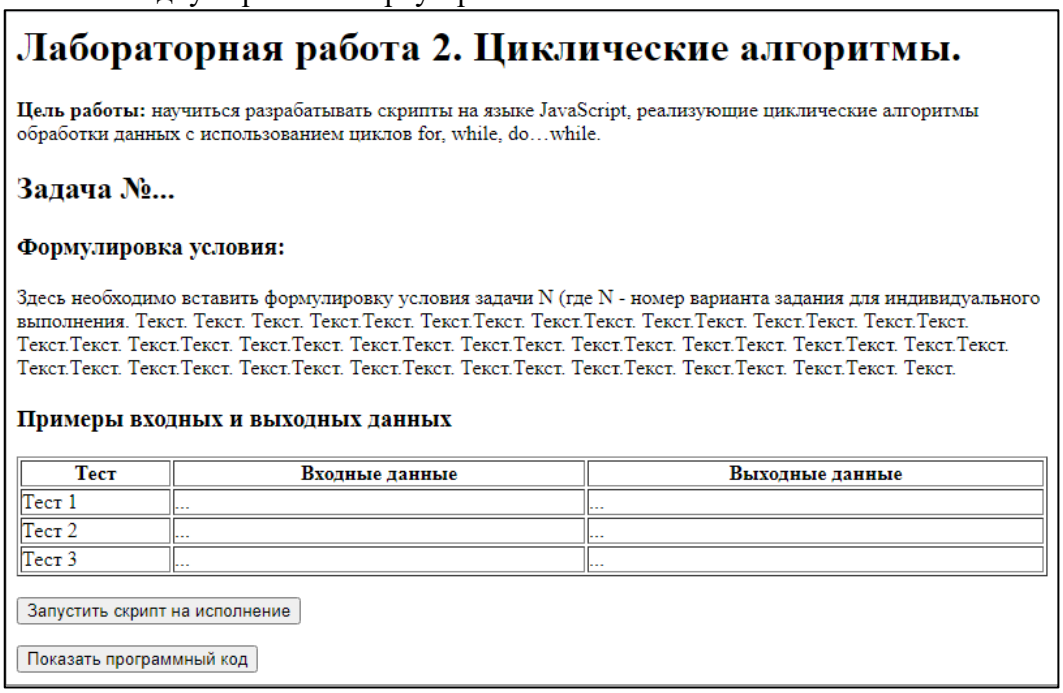
Задания для выполнения

Задание 1. Разработайте скрипт, реализующий циклический алгоритм решения задачи с использованием цикла `for`. Ввод исходных данных осуществлять с использованием функции `promt`. Вывод полученных результатов осуществлять с использованием функции `alert`. Сохраните скрипт в файле **script_2_1.js**.

Задание 2. Разработайте скрипт, реализующий циклический алгоритм решения задачи с использованием цикла `while`. Ввод исходных данных осуществлять с использованием функции `promt`. Вывод полученных результатов осуществлять с использованием функции `alert`. Сохраните скрипт в файле **script_2_2.js**

Задание 3. Разработайте скрипт, реализующий циклический алгоритм решения задачи с использованием цикла do…`while`. Ввод исходных данных осуществлять с использованием функции `promt`. Вывод полученных результатов осуществлять с использованием функции `alert`. Сохраните скрипт в файле **script_2_3.js**

Задание 4. Создайте по образцу html-документы **lab_2_1.html**, **lab_2_2.html** и **lab_2_3.html** и подключите к ним внешние скрипты **script_2_1.js**, **script_2_2.js** и **script_2_3.js** соответственно. Для тестирования работоспособности каждого скрипта разработайте по 3 теста (включающих наборы входных и выходных данных). Выполните тестирование и отладку скриптов в браузере.



Лабораторная работа 2. Циклические алгоритмы.

Цель работы: научиться разрабатывать скрипты на языке JavaScript, реализующие циклические алгоритмы обработки данных с использованием циклов for, while, do…while.

Задача №...

Формулировка условия:

Здесь необходимо вставить формулировку условия задачи N (где N - номер варианта задания для индивидуального выполнения. Текст. Текст. Текст. Текст.

Примеры входных и выходных данных

Тест	Входные данные	Выходные данные
Тест 1	...	...
Тест 2	...	...
Тест 3	...	...

Запустить скрипт на исполнение

Показать программный код

Образец html-документа

Варианты индивидуальных задач к Заданию 1:

1. Алгоритм Евклида нахождения наибольшего общего делителя (НОД) неотрицательных целых чисел основан на следующих свойствах этой величины. Пусть *m* и *n* — одновременно не равные нулю целые неотрицательные числа и пусть $m \ge n$. Тогда, если *n*=0, то НОД(*n*, *m*) = *m*, а если $n \ne 0$, то для чисел *n*, *m* и *r*, где *r* — остаток от деления *m* на *n*, выполняется равенство НОД(*n*, *m*) = НОД(*n*, *r*). Например, НОД(15, 6) = НОД(6, 3) = НОД (3, 0) = 3. Даны натуральные числа *n*, *m*. Используя алгоритм Евклида, найти их наибольший общий делитель.
2. Дано действительное число ε ($\varepsilon > 0$). Вычислить $\sum_{k=1}^{\infty} \frac{1}{k^2}$ со степенью точности ε (считать, что требуемая степень точности достигнута, если вычислена сумма нескольких первых слагаемых и очередное слагаемое оказалось по модулю меньше, чем ε, — это и все последующие слагаемые можно уже не учитывать).
3. Найти наибольшее положительное целое число *n*, удовлетворяющее следующему условию: $3n^5 - 730n < 5$.
4. Вычислить значения функции $y = 7x^3 - 3x^2 + 15$ для значений *x*, изменяющихся от -1 до 2 с шагом 0.2.
5. Найти значение минимального положительного члена числовой последовательности, заданной следующими соотношениями: $x_n = x_{n-1} + x_{n-2} + 100$, $x_1 = x_2 = -99$.
6. Дано действительное число *x*. Вычислить $\sum_{k=1}^{\infty} \frac{1}{x^3 k^2}$ с точностью 10^{-7}.
7. Найти *n*-ный член (*n*>2) числовой последовательности, заданной рекуррентными соотношениями: $x_n = x_{n-1} + x_{n-2}$, $x_0 = x_1 = 1$.
8. Цилиндр единичного объема имеет высоту *h*. Определить радиус основания цилиндра для значений *h*, равных 0.5, 1, 1.5, ... , 7.
9. Даны действительные числа *x*, ε ($\varepsilon > 0$).Найти первый член a_n последовательности, заданной следующими соотношениями: $a_k = \sqrt{|5a_{k-1} - 2x|}$, $a_1 = x$, *k* = 2, 3, ..., для которого выполняется условие $|a_n - a_{n-1}| < \varepsilon$.
10. Вычислить значения функции $y = \frac{1}{4}\left(\frac{1}{a^4} - \frac{1}{x^4}\right) arctg\frac{x}{a} - \frac{1}{12ax^3} + \frac{1}{4a^3 x}$ для значений *x*, изменяющихся от 1 до 1.6 с шагом 0.05; *a* — целое число.
11. Можно ли заданное натуральное число представить в виде суммы двух квадратов натуральных чисел?
12. Пусть $x_1 = 0.2$, $x_2 = -0.2$, $x_n = n + \sin x_{n-2}$, *n* = 3, 4, ... Среди x_1, ... , x_{100} найти ближайшее к какому-нибудь целому.
13. Дано неравенство $-4n^4 + 841\sqrt{n} + 3 \ge 0$. Найти наибольшее натуральное число *n*, при котором это неравенство было бы истинным.
14. Пусть $x_0 = 1$, $x_k = \frac{2 - x_{k-1}^3}{7}$. Найти первый член x_n, для которого $|x_n - x_{n-1}| < 10^{-5}$.

15. Вычислить $(1+\sin 0.1)(1+\sin 0.2)(1+\sin 0.3)\ldots(1+\sin 10)$.

16. Найти n-ный член (n>2) числовой последовательности, заданной соотношениями: $x_n = 2x_{n-1} - 3x_{n-2}$, $x_0 = 0,\ x_1 = 9$.

17. Дано действительное число x. Вычислить с точностью 10^{-6} бесконечную сумму $\sum_{k=1}^{\infty} \frac{1}{x^2 + k^3}$.

18. Найти значение минимального положительного члена числовой последовательности, заданной рекуррентными соотношениями: $x_n = x_{n-1} + x_{n-2} + x_{n-3} + 200$, $x_1 = x_2 = x_3 = -99$.

19. Дано действительное число ε ($\varepsilon > 0$). Вычислить $\sum_{i=1}^{\infty} \frac{1}{i(i+1)}$ с точностью ε.

20. Вычислить значения функции $y = 2\ln x - 3x^2 + 1$ на отрезке $[a,b]$ с шагом h (a, h >0).

21. Вычислить значения функции $y = ae^{-bx}$ для x = 0.1, 0.2, 0.3, ..., прекратив вычисления тогда, когда значение e^{-bx} станет меньше 0.001; a и b — действительные числа.

22. Дано действительное число a>0. Последовательность $x_0,\ x_1,\ \ldots$ образована по закону

$$x_0 = \begin{cases} a, & \text{при } a \le 1, \\ \frac{a}{5}, & \text{при } 1 < a < 25, \\ \frac{a}{25}, & \text{при } a \ge 25, \end{cases} \quad x_n = \frac{3}{7} x_{n-1} + \frac{a}{5x_{n-1}^3},$$ где n = 1, 2, ... Найти член x_n.

23. Получить таблицу температур по Цельсию от 0 до 100 градусов и их эквивалентов по шкале Фаренгейта, если известно, что $t_F = \frac{9}{5} t_C + 32$.

24. Даны действительные числа x и ε (ε >0). Вычислить $\sum \frac{1}{\sqrt{|x|} + k^2}$ с точностью ε.

25. Вычислить значения многочлена $x^5 - 9x^4 + 1.7x^2 - 9.6$ для x = 0, 0.5, 1, ..., 5.

26. Найти наибольшее положительное целое число n, удовлетворяющее условию $7n^3 + 81n^2 - 10^6 < 0$.

27. Даны действительные числа x, ε (ε >0). Последовательность $a_1,\ a_2,\ \ldots$ образована по следующему закону: $a_1 = x$, $a_n = \frac{a_{n-1}}{n+1}$ (n = 2, 3, ...). Найти первый член a_n, такой, что $|a_n - a_{n-1}| < \varepsilon$.

28. Вычислить сумму $\sum_{n=1}^{\infty} \frac{n}{n^3 + n^2}$ с точностью 10^{-6}.

29. Даны положительные действительные числа *a, x, ε*. В последовательности y_1, y_2, ..., образованной по закону $y_0 = a$; $y_i = \frac{1}{2}\left(y_{i-1} + \frac{x}{y_{i-1}}\right)$, *i* = 1, 2, ... , найти первый член y_n, для которого справедливо неравенство $|y_n - y_{n-1}| < \varepsilon$.

30. Член ряда с номером *n* определяется выражением $\frac{n}{n^3 + n^2 - 3}$. Вычислить сумму членов ряда от первого до члена с наименьшим номером, не превышающего по абсолютному значению 0.01.

Варианты индивидуальных задач к Заданиям 2 и 3:

1. Вычислить значения величины $q = e^{-\frac{k}{2}}\left(k - 3\sqrt{k} + 1{,}2\right)$, где *k* = 0.1, 0.2, ... Расчет производить до тех пор, пока *q* не станет меньше 0.0001.
2. Даны действительные числа *a, b* (*a*<*b*), натуральное число *n*, функция $y = x|x+1|$, определенная на отрезке [*a, b*]. Вычислить значения этой функции для значений аргумента $x_i = a + ih$ (*i* = 0, 1, ..., *n*), $h = \frac{b-a}{n}$.
3. Дано действительное число ε (ε >0). Последовательность a_1, a_2, ... образована по следующему закону: $a_n = \frac{n}{\sqrt{n^2+1} + \sqrt{n^2-1}}$. Найти первый член a_n, для которого выполнено условие: $|a_n - a_{n-1}| < \varepsilon$.
4. Вычислить *n*-ный член числовой последовательности, заданной рекуррентными соотношениями: $x_n = x_{n-1} + x_{n-2} + x_{n-3}$; $x_0 = x_1 = 1$, $x_2 = 6$.
5. Вычислить значения функции $y = x \cdot tg\frac{x}{2} + 2\ln\left|\cos\frac{x}{2}\right|$, где *x* = 0.5, 0.6, 0.7, ..., 1,5.
6. Пусть $y_0 = 0$; $y_k = \frac{y_{k-1} + 1}{y_{k-1} + 2}$, *k* = 1, 2, ... Дано действительное ε >0. Найти первый член y_n, для которого выполняется неравенство $|y_n - y_{n-1}| < \varepsilon$.
7. Найти значения полинома $y = 8.0m + 4.3m^2 - 1.46m^3$ для $1.0 \le m \le 5.9$ с шагом $h_m = 0.1$.
8. Вычислить наибольшее положительное число *n*, удовлетворяющее условию $e^n - 1000\lg n \le 5$.
9. Член ряда с номером *n* определяется выражением $\frac{e^{2n}\sin e^n}{n^2}$. Найти сумму членов ряда от первого до члена с наименьшим номером, не превышающим по абсолютной величине 10^{-5}.
10. Дано натуральное число *n*. Вычислить значения функции $y = \frac{x^2 - 3x + 2}{\sqrt{2x^3 - 1}}$ для *x* = 1, 1.1, 1.2, ..., 1+0.1 *n*.

11. Даны действительные числа *x*, ε (ε >0). Последовательность a_1, a_2, ... образована по следующему закону: $a_1 = x$, $a_n = \frac{1}{2a_{n-1}} + \frac{x}{4 + a_{n-1}^2}$ (*n* = 2, 3, ...). Найти первый член a_n, такой, что $|a_n - a_{n-1}| < \varepsilon$.

12. Вычислить значения функции $f(x) = \frac{x}{e^x - 1}$, если аргумент меняется от 0.1 до 2.0 с шагом 0.02.

13. Дано действительное число ε (ε >0). Вычислить с точностью ε значение $\sum_{k=1}^{\infty} e^{-k^2}$.

14. Даны натуральное число *n*, действительные числа *a*, *b* (*a*<*b*). Получить r_0, r_1, ..., r_n, где $r_i = a + ih$, $h = \frac{b-a}{n}$.

15. Вычислить последовательности значений функций $p_1(x) = x$, $p_2(x) = \frac{3x^2 - 1}{2}$, $p_3(x) = \frac{5x^2 - 3x}{2}$ для значений аргумента x = 0, 0.05, 0.1, ..., 20.

16. Найти значение минимального положительного члена числовой последовательности, заданной следующими соотношениями: $x_n = x_{n-1} + x_{n-3} + 100$, $x_1 = x_2 = x_3 = -99$.

17. Вычислить *n*-ный член числовой последовательности, заданной рекуррентными соотношениями: $x_n = x_{n-1} + 4x_{n-3}$; $x_0 = x_1 = x_2 = 2$.

18. Дано действительное число *x*. Вычислить $\sum_{k=1}^{\infty} \frac{\sqrt{|x|}}{k^3}$ с точностью 10^{-6}.

19. Найти значения функции $y = \sqrt{(\pi \cdot a)^2 + \frac{1}{7} + e^{-\frac{a}{2}}}$, если *a* меняется от 1.3 до 2.8 с шагом 0.1.

20. Даны действительные числа *x*, ε (ε >0). Последовательность a_1, a_2, ... образована по следующему закону: $a_1 = x$, $a_n = 3 + \frac{1}{e^n}\cos^2(a_{n-1} - x)$ (*n* = 2, 3, ...). Найти первый член a_n, такой, что $|a_n - a_{n-1}| < \varepsilon$.

21. Дано действительное число *x*. Вычислить $\sum_{k=1}^{\infty} \frac{x}{k^3 + k\sqrt{|x|} + 1}$ с точностью 10^{-6}.

22. Вычислить наибольшее положительное число *n*, удовлетворяющее условию $-e^n + 30n^2 - 1 > 0$.

23. Даны действительные числа *a*, *b*, ε (*a*>*b*>0, ε >0). Последовательности x_1, x_2, ..., y_1, y_2, ... образованы по закону: $x_1 = a$, $y_1 = b$, $x_k = \frac{1}{2}(x_{k-1} + y_{k-1})$, $y_k = \sqrt{x_{k-1} y_{k-1}}$. Найти первое x_n, такое, что $|x_n - y_n| < \varepsilon$.

24. Даны действительные числа *a, b*. Вычислить значения функции $y = \frac{1}{a\sqrt{a^2+b^2}} arctg \frac{a \cdot tgx}{\sqrt{a^2+b^2}}$, если *x* = 0.1, 0.3, 0.5, ..., 1.5.

25. Вычислить *n*-ный член числовой последовательности, заданной следующими соотношениями: $x_n = x_{n-1}\left(x_{n-2}+1\right);\ x_0 = 0,\ x_1 = 1$.

26. Найти значения функции $y = x - tgx$ для значений *x* = 1.1, 1.2, 1.3, ..., 1.9.

27. Дано действительное число ε (ε >0). Вычислить $\sum_{n=1}^{\infty} \frac{1}{e^n} \cos^2\left(e^n - 1\right)$, учитывая только те слагаемые, в которых множитель $\frac{1}{e^n}$ имеет величину, не меньшую, чем ε.

28. Даны действительные числа *a, b*. Последовательности x_1, x_2, ..., y_1, y_2, ... образованы по следующему закону: $x_n = a + b\cos(0.5n)$, $y_n = 0.5an - b\sin(0.5n)$. Получить $\frac{x_k}{y_k}$, где *k* — наибольшее натуральное число, удовлетворяющее двум условиям: $k \le 20$ и $\left|y_k\right| > 10^{-3}$.

29. Вычислить функцию $y = x - \frac{3}{x^3} + \frac{5}{x^5} - \frac{7}{x^7} + \ldots$ Вычисления прекратить, когда очередной член будет по абсолютному значению меньше 10^{-6}.

30. Дано действительное число ε (ε >0). Вычислить сумму $\sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{i(i+1)(i+2)}$ с точностью ε. Считать, что требуемая точность достигнута, если вычислена сумма нескольких первых слагаемых и очередное слагаемое оказалось по модулю меньше, чем ε, — это и все последующие слагаемые можно уже не учитывать.

Лабораторная работа 3. Функции. Рекурсивные функции

Цель работы: научиться разрабатывать скрипты на языке JavaScript с использованием функций и рекурсивных функций.

Теоретический материал по теме

Функция - это программный код, который определяется один раз и затем может вызываться на выполнение любое количество раз. В JavaScript **функция** - специальный тип объектов, позволяющий формализовать средствами языка определённую логику поведения и обработки данных.

Объявление функции осуществляется следующим образом:

```
function имя(параметры){
    код функции
}
```

Имя - обязательный идентификатор, определяющий **имя функции**. Имя функции может содержать буквы, цифры (0-9), знаки «$» и «_»; в качестве букв рекомендуется использовать только буквы английского алфавита (a-z, A-Z); имя функции, также как и имя переменной, не может начинаться с цифры.

В качестве имени функции обычно выбирают глагол, т. к. функция выполняет действие. Оно должно быть простым, точным и описывать действие функции, чтобы программист, который будет читать код, получил верное представление о том, что делает функция. Как правило, используются глагольные префиксы, обозначающие общий характер действия, после которых следует уточнение. Обычно в командах разработчиков действуют соглашения, касающиеся значений этих префиксов. Например, функции, начинающиеся с…
"show…" – что-то показывают;
"get…" – возвращают значение;
"calc…" – что-то вычисляют;
"create…" – что-то создают;
"check…" – что-то проверяют и возвращают логическое значение, и т.д.

Параметры функции - это локальные переменные функции, которые определяются на этапе объявления функции в круглых скобках. Обратиться к параметрам можно только внутри функции, снаружи они не доступны. Параметров может быть указано любое количество; если параметров несколько, то их между собой необходимо разделить посредством запятой. Параметры позволяют более удобно (по имени) получить переданные аргументы функции при её вызове.

При вызове функции, ей могут передаваться значения, которыми будут инициализированы параметры. Значения, которые передаются при вызове функции, называются **аргументами**. Аргументы, указываются через запятую. Когда при вызове функции ей передаётся список аргументов, эти аргументы присваиваются параметрам функции в том порядке, в каком они указаны: первый аргумент присваивается первому параметру, второй аргумент – второму параметру и т. д.

Код функции - это набор инструкций, заключенный в фигурные скобки, которые необходимо выполнить при её вызове. Код функции называют ещё её **телом**. Тело функции может быть пустым, но фигурные скобки должны быть указаны всегда.

С помощью инструкции `return` функция может возвратить некоторое значение (результат работы функции) программе, которая её вызвала. Возвращаемое значение передаётся в точку вызова функции. Инструкция `return` имеет следующий синтаксис:
return выражение;

В программу возвращается не само выражение, а результат его вычисления. Инструкция `return` может быть расположена в любом месте функции. Как только будет достигнута инструкция `return`, функция возвращает значение и немедленно завершает своё выполнение. Код, расположенный после инструкции `return`, будет проигнорирован. Если инструкция `return` не указана или не указано возвращаемое значение, то функция вернёт значение `undefined`.

Примеры:

```
1)
function sum(a, b) {
  var s = a + b;
  alert(s);
}
sum(5, 7);   // 12

2)
function calc(a) {
  return a * a;
}
var x = calc(5);
```

```
alert(x);   // 25
```

3)
```
function check(a, b) {
  if(a > b) return a;
  else return b;
}
alert(check(3, 5));   // 5
```

Рекурсивно определенная функция - это функция, которая определяется в терминах более простой версии самого себя. **Рекурсия** – это приём программирования, полезный в ситуациях, когда задача может быть естественно разделена на несколько аналогичных, но более простых задач. Или когда задача может быть упрощена до несложных действий плюс простой вариант той же задачи.

Рекурсивная функция обязательно должна иметь условие завершения, если его не указать, функция будет вызываться до тех пор, пока не будет достигнута максимальная глубина рекурсии, затем будет сгенерировано исключение.Общее количество вложенных вызовов называют **глубиной рекурсии**. Максимальная глубина рекурсии в браузерах ограничена 10 000 рекурсивных вызовов.

Пример рекурсивной функции, определяющей факториал числа n!:
```
function getFactorial(n){
    if (n === 1)return 1;
    else return n*getFactorial(n-1);
}
var result = getFactorial(4);
alert(result); // 24
```

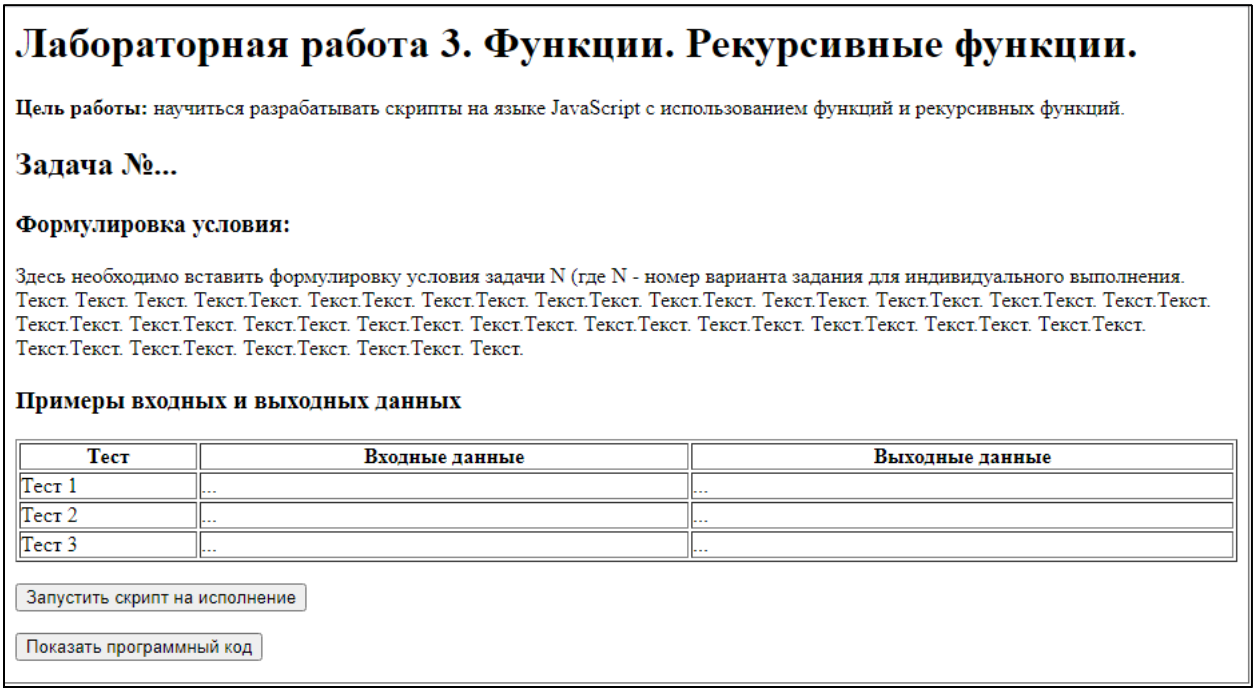
Функция `getFactorial()` возвращает значение 1, если параметр n равен 1, либо возвращает в качестве результата произведение параметра n на значение той же функции `getFactorial`, в которую передается значение n-1. Например, при передаче числа 4, у нас образуется следующая цепочка вызовов:
```
var result = 4 * getFactorial(3);
var result = 4 * 3 * getFactorial(2);
var result = 4 * 3 * 2 * getFactorial(1);
var result = 4 * 3 * 2 * 1; // 24
```

Задания для выполнения

Задание 1. Разработайте скрипт, реализующий алгоритм решения задачи с использованием функции. Ввод исходных данных осуществлять с использованием функции `promt`. Вывод полученных результатов осуществлять с использованием функции `alert`. Сохраните скрипт в файле **script_3_1.js**.

Задание 2. Разработайте скрипт, реализующий алгоритм решения задачи с использованием рекурсивной функции. Ввод исходных данных осуществлять с использованием функции `promt`. Вывод полученных результатов осуществлять с использованием функции `alert`. Сохраните скрипт в файле **script_3_2.js**

Задание 3. Создайте по образцу html-документы **lab_3_1.html** и **lab_3_2.html** и подключите к ним внешние скрипты **script_3_1.js** и **script_3_2.js** соответственно. Для тестирования работоспособности каждого скрипта разработайте по 3 теста (включающих наборы входных и выходных данных). Выполните тестирование и отладку скриптов в браузере.



Лабораторная работа 3. Функции. Рекурсивные функции.

Цель работы: научиться разрабатывать скрипты на языке JavaScript с использованием функций и рекурсивных функций.

Задача №...

Формулировка условия:

Здесь необходимо вставить формулировку условия задачи N (где N - номер варианта задания для индивидуального выполнения. Текст. Текст. Текст. Текст.

Примеры входных и выходных данных

Тест	Входные данные	Выходные данные
Тест 1	...	...
Тест 2	...	...
Тест 3	...	...

Запустить скрипт на исполнение

Показать программный код

Образец html-документа

Варианты индивидуальных задач к Заданию 1:

1. Написать и протестировать функцию, которая вычисляет сумму квадратов *n* ее аргументов, где $1 \le n \le 3$.
2. Даны действительные числа *x*, *y* и *z*. Получить $u = \dfrac{\max(x, y) + \max(x + y, z)}{(\max(0.5, x + z))^2}$.
3. Даны действительные числа *s*, *t*. Получить $f(t,\ -2s,\ 1.17) + f(2.2,\ t,\ s - t)$, где $f(a, b, c) = \dfrac{2a - b - \sin c}{5 + |c|}$.
4. Даны действительные числа *a*, *b*, *c*. Получить $\dfrac{\max(a,\ a + b) + \max(a,\ b + c)}{1 + \max(a + bc,\ 1,\ 1.5)}$.
5. Написать и протестировать функцию, определяющую периметр плоского *n*-угольника, координаты вершин которого передаются в качестве параметра в виде 2*n* чисел: x_i и y_i.
6. Написать и протестировать функцию и именем COSIN, вычисляющую *cos*(*x* + *iy*) по формуле $\cos(x + iy) = \cos x \cdot \dfrac{e^y + e^{-y}}{2} \cdot \sin x \cdot \dfrac{e^y - e^{-y}}{2} i$, где *x* и *y* — два параметра вещественного типа.
7. Даны действительные числа *a* и *b*. Получить $u = f(0.5,\ a) + f(a,\ b) + f(a + b,\ a - b)$, где $f(x, y) = \dfrac{x^2 + xy - y^2}{1 + x^2 + y^2}$.

8. Написать и протестировать функцию, которая возвращает следующие значения: $y = \begin{cases} 1+\sqrt{1+x^2}, & \textit{если} \quad x<0, \\ 0, & \textit{если} \quad x=0, \\ 1-\sqrt{1+x^2}, & \textit{если} \quad x>0. \end{cases}$

9. Написать и протестировать функцию и именем *EXPONENT*, вычисляющую $e^{x+iy} = e^x\left(\cos y + i\sin y\right)$, где *x* и *y* — два параметра вещественного типа.

10. Дано действительное число *y*. Получить $\dfrac{1.7t(0.25)+2t(1+y)}{6-t\left(y^2-1\right)}$, где $t(x) = \dfrac{\sum\limits_{k=0}^{10}\dfrac{e^{2k+1}}{(2k+1)!}}{\sum\limits_{k=0}^{10}\dfrac{e^{2k}}{(2k)!}}$.

11. Написать и протестировать логическую функцию, которая определяет, принадлежит ли заданная точка к внутренней области круга $\left(x-x_0\right)^2+\left(y-y_0\right)^2 \le R^2$. Формальные параметры: x_0, y_0 — координаты центра круга, *R* — радиус круга, *x*, *y* — координаты проверяемой точки.

12. Даны действительные числа *m* и *n*. Получить $l = \dfrac{m!+n!}{(m+n)!}$, введя в употребление функцию $fact(n) = 1\cdot 2\cdot\ldots\cdot n$.

13. Даны натуральное число *n*, действительные числа *s*, *t*, a_0, …, a_n. Получить $p(1)-p(t)+p^2(s-t)-p^3(1)$, где $p(x) = a_n e^{nx} + a_{n-1}e^{(n-1)x} + \ldots + a_0$.

14. Написать и протестировать функцию, вычисляющую расстояние между двумя точками в *n*-мерном евклидовом пространстве, где $1 \le n \le 3$. Координаты точек передаются в виде 2*n* отдельных параметров.

15. Даны натуральные числа *k*, *l*, *m*, действительные числа x_1, …, x_k, y_1, …, y_l, z_1, …, z_m. Получить:

$$t = \begin{cases} \left(\max\left(y_1,\ \ldots,\ y_l\right)+\max\left(z_1,\ \ldots,\ z_m\right)\right) & \textit{при} \quad \max\left(x_1,\ \ldots,\ x_k\right) \ge 0, \\ 1+\left(\max\left(x_1,\ \ldots,\ x_k\right)\right)^2 & \textit{при} \quad \max\left(x_1,\ \ldots,\ x_k\right) < 0. \end{cases}$$

16. Написать и протестировать функцию для вычисления *t* с формальным параметром *x*. $f = -0.5 + 1.25x - 3.25x^2 - 1.87x^3 + 0.5x^4$. Формальный параметр и значение функции должны быть действительного типа.

17. Даны действительные числа *a*, *b*. Получить *u* = min(*a*, *b*), *v* = min(*ab*, *a*+*b*), min(*u*+*v*, 3.25).

18. Написать и протестировать функцию для вычисления *H* с формальными параметрами *x*, *y* и *z*: $H = (x+4)^3 - \dfrac{y}{2} + x\ln z$. Формальные параметры и значение функции должны быть действительного типа.

19. Даны действительные числа *s*, *t*. Получить $h(s,t)+\max\left(h^2\left(s-t,\ st\right),\ h^4\left(s-t,\ s+t\right)\right)+h(1,\ 1)$, где $h(a,b)=\dfrac{a}{1+b^2}+\dfrac{b}{1+a^2}-ab$.

20. Написать и протестировать функцию для вычисления *g* с формальными параметрами *x*, *y* $g=e^{-(x-1)^2}-y(x-1)^2$. Формальные параметры и значение функции должны быть действительного типа.

21. Даны действительные числа *s*, *t*. Получить $g(1{,}2,\ s)+g(t,\ s)-g(2s-1,\ st)$, где $g(a,\ b)=\dfrac{a^2+b^2}{a^2+2ab+3b^2+4}$.

22. Написать и протестировать функцию, позволяющую найти корень *m*-той степени из *x*, применив тождество: $\sqrt[m]{x}=e^{\frac{\ln x}{m}}$.

23. Написать и протестировать функцию для вычисления *I* с формальным параметром *x*: $I=e^{M}-e^{-M}$, где $M=x^3+0.5x^2-x$.

24. Написать и протестировать функцию *f*, которая возвращала бы среднее геометрическое n ее параметров по формуле $f=\sqrt[n]{x_1\cdot x_2\cdot\ldots\cdot x_n}$.

25. Написать и протестировать функцию, которая одной действительной переменной присваивает значение, равное сумме квадратов двух действительных значений, а другой переменной — значение, равное удвоенному произведению тех же двух действительных значений.

26. Написать и протестировать процедуру для определения целой части и остатка кубического корня из натурального числа *x* ($\sqrt[3]{x}=e^{\frac{\ln x}{3}}$).

27. Написать и протестировать функцию, которая вычисляет среднее арифметическое, среднее геометрическое и среднее гармоническое двух чисел a_1 и a_2 по следующим формулам: $a_{ар}=\dfrac{a_1+a_2}{2}$, $a_{геом}=\sqrt{a_1\cdot a_2}$, $a_{гарм}=\dfrac{2}{\frac{1}{a_1}+\frac{1}{a_2}}$.

28. Написать и протестировать функцию нахождения координат *(x, y)* точки, делящей отрезок *AB* пополам, если известно, что его концы имеют координаты: $A(x_1, y_1)$, $B(x_2, y_2)$.

29. Даны два действительных числа *m* и *n* ($n\neq0$). Написать и протестировать функцию для определения целой части и остатка от деления *m* на *n*.

30. Написать и протестировать функцию нахождения минимума и максимума из 4 действительных значений.

Варианты индивидуальных задач к Заданию 2:

1. Дано натуральное число *n*; найти количество его цифр. Использовать программу, включающую рекурсивную функцию.

2. Числа Фибоначчи u_0, u_1, u_2 определяются следующим образом: $u_0=0$, $u_1=1$, $u_n=u_{n-1}+u_{n-2}$ (*n* = 2, 3, ...). Написать программу вычисления u_n для данного неотрицательного целого *n*, включающую рекурсивную функцию, которая основана на непосредственном использовании соотношения $u_n=u_{n-1}+u_{n-2}$.

3. При положительном *a* решением уравнения $x = \frac{2x}{3} + \frac{a}{3x^2}$ служит $x = \sqrt[3]{a}$. Рекуррентное соотношение $x_i = \frac{2x_{i-1}}{3} + \frac{a}{3x_{i-1}^2}$, $x_1 = 1$ можно использовать для быстрого вычисления $\sqrt[3]{a}$, так как элементы последовательности при увеличении *i* очень быстро приближаются к $\sqrt[3]{a}$. Написать программу вычисления n-ного члена указанной выше последовательности.

4. Написать и протестировать рекурсивную функцию *pow*(*x*, *n*) от вещественного *x* (*x*≠0) и целого *n*, которая вычисляет величину x^n согласно формуле
$$x^n = \begin{cases} 1 & при \quad n = 0, \\ 1/x^{|n|} & при \quad n < 0, \\ x \cdot x^{n-1} & при \quad n > 0. \end{cases}$$

5. Последовательность чисел задана соотношениями: $a_1 = 2m + n$, $a_i = 2m + \frac{a_{i-1} \cdot n}{a_{i-1} + n}$. Написать и протестировать рекурсивную функцию вычисления *n*-ного члена этой последовательности.

6. Дано натуральное число *n*; найти *n*!. Использовать программу, включающую рекурсивную функцию вычисления *n*!.

7. Написать рекурсивную функцию вычисления суммы $1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!}$. Обозначьте для этого $a_i = \frac{x^i}{i!}$ и используйте следующие рекурсивные соотношения: $a_i = a_{i-1} \cdot \frac{x}{i}$, $a_0 = 1$, $s_i = s_{i-1} + a_i$, $s_0 = 1$.

8. Вычислить n-ный член последовательности, заданной следующими соотношениями: $a_1 = 2$, $a_i = 2a_{i-1} - 1$. Использовать программу, включающую рекурсивную функцию.

9. Арифметической прогрессией называется числовая последовательность, каждый член которой, начиная со второго, равен предыдущему, сложенному с некоторым числом. Написать и протестировать рекурсивную функцию вычисления n-ного члена арифметической прогрессии.

10. Написать и протестировать рекурсивную функцию C(m, n), где 0 ≤ m ≤ n, для вычисления биноминального коэффициента C_n^m по следующей формуле: $C_n^0 = C_n^n = 1$; $C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$ при 0 < m < n.

11. Вычислить n-ные члены последовательностей x_1, x_2, … и y_1, y_2, …, если они заданы следующими соотношениями: $x_1 = 1$, $x_i = x_{i-1} + 0.1 \cdot y_{i-1}$; $y_1 = 0$, $y_i = y_{i-1} + 0.1 \cdot x_{i-1}$.

12. Даны натуральные числа n, m, причем $n \ge m$; найти НОД(n, m). Использовать программу, включающую рекурсивную функцию вычисления НОД, основанную на соотношении НОД(n, m) = НОД(m, r), где r — остаток от деления n на m?

13. Написать и протестировать рекурсивную функцию для вычисления N-ного члена последовательности, начинающейся с единицы, в которой каждый следующий член равен сумме обратных величин всех предыдущих.

14. Функция f(n) для целых неотрицательных n определена следующим образом: $f(n) = \begin{cases} n-10, & \text{если } n > 100, \\ f\big(f(n+11)\big), & \text{если } n \le 100 \end{cases}$. Вычислить f(106), f(99), f(85). Какие вообще значения принимает эта функция?

15. Написать и протестировать рекурсивную функцию для вычисления N-ного члена последовательности, начинающейся с единицы, в которой каждый следующий член равен сумме квадратов всех предыдущих.

16. Даны натуральные числа a, c, m. Получить f(m), где $f(n) = \begin{cases} n, & \text{если } 0 \le n \le 9, \\ g(n) f\big(n-1-g(n)\big), & \text{если } n < 0 \text{ или } n > 9 \end{cases}$. Здесь g(n) — остаток от деления $an + c$ на 10. Использовать программу, включающую рекурсивную функцию вычисления f(n).

17. Геометрической прогрессией называется числовая последовательность, каждый член которой, начиная со второго, равен предыдущему, умноженному на некоторое отличное от нуля постоянное число. Написать и протестировать рекурсивную функцию вычисления n-ного члена геометрической прогрессии.

18. Написать и протестировать рекурсивную функцию root(f, a, b, eps), которая методом деления отрезка пополам находит с точностью eps корень уравнения f(x)=0 на отрезке [a, b] (Считать, что eps>0, f(a)f(b)<0 и f(x) — непрерывная и монотонная функция на отрезке [a, b]).

19. Даны неотрицательные целые числа n, m; вычислить A(n, m), где $A(n,\ m) = \begin{cases} m+1, & \text{если } n = 0, \\ A(n-1,\ 1), & \text{если } n \ne 0,\ m = 0, \\ A\big(n-1,\ A(n,\ m-1)\big), & \text{если } n > 0,\ m > 0 \end{cases}$ (это — так называемая функция Аккермана). Использовать программу, включающую рекурсивную функцию.

20. Написать и протестировать функцию min(x) для определения минимального элемента одномерного числового массива x, введя вспомогательную рекурсивную функцию min(k), находящую минимум среди последних элементов массива x, начиная с k-го.

21. Функция f(n) для целых неотрицательных n определена следующим образом: $f(0) = 0$, $f(1) = 1$, $f(2n) = f(n)$, $f(2n+1) = f(n) + f(n+1)$. Для данного N найти и напечатать f(N).

22. Написать и протестировать рекурсивную логическую функцию symm(s, i, j), проверяющую, является ли симметричной часть строки s, начинающаяся i-м и кончающаяся j-м ее элементами.

23. С клавиатуры вводится последовательность положительных вещественных чисел, за которой следует отрицательное число. Написать и протестировать рекурсивную функцию sum без параметров для нахождения суммы этих положительных чисел.
24. Написать и протестировать рекурсивную функцию для вычисления N-ного члена последовательности, начинающейся с единицы, в которой каждый следующий член равен синусу суммы всех предыдущих.
25. При положительном a решением уравнения $x = \frac{x}{2} + \frac{a}{2x}$ служит $x = \sqrt{a}$. Рекуррентное соотношение $x_i = \frac{x_{i-1}}{2} + \frac{a}{2x_{i-1}}$, $x_1 = 1$ можно использовать для быстрого вычисления $\sqrt{a}$, так как элементы последовательности при увеличении i очень быстро приближаются к $\sqrt{a}$.Написать программу вычисления n-ного члена указанной выше последовательности.
26. Написать и протестировать рекурсивную функцию для вычисления n-ного члена последовательности, заданной следующими соотношениями: $a_i = i \cdot a_{i-1} + a_{i-2}$, $a_1 = a_2 = 1$.
27. Дано натуральное число n; найти сумму его цифр. Использовать программу, включающую рекурсивную или функцию.
28. Написать и протестировать рекурсивную процедуру или функцию для вычисления n-ного члена последовательности, заданной следующими соотношениями: $a_i = a_{i-1} + a_{i-2} + a_{i-3}$, $a_1 = a_2 = a_3 = 1$.
29. Написать и протестировать функцию max(x) для определения максимального элемента одномерного числового массива x, введя вспомогательную рекурсивную функцию max(k), находящую максимум среди последних элементов массива x, начиная с k-го.
30. Дано натуральное число n; найти произведение его цифр. Использовать программу, включающую рекурсивную функцию.

Лабораторная работа 4. Алгоритмы работы со строками

Цель работы: научиться разрабатывать скрипты на языке JavaScript, реализующие алгоритмы обработки строковых данных с использованием свойств и методов объекта String.

Теоретический материал по теме

Строковый тип данных (string) предназначен для обработки строк, т.е. последовательностей символов, заключенных в одинарные или двойные кавычки.

Строковый литерал – это последовательность из нуля или более Unicode-символов, заключенная в одинарные или двойные кавычки.

Примеры строк:

```
"" // Это пустая строка: в ней ноль символов
'testing'
"3.14"
'name="myform"'
```

```
"Пи - это отношение длины окружности круга к его диаметру"
```

Сами символы двойных кавычек могут содержаться в строках, ограниченных символами одинарных кавычек, а символы одинарных кавычек – в строках, ограниченных символами двойных кавычек. Строковые литералы должны записываться в одной строке программы и не могут разбиваться на две строки.

Программы на JavaScript часто содержат строки HTML-кода, а HTML-код, в свою очередь, часто содержит строки JavaScript-кода. Как и в JavaScript, в HTML для ограничения строк применяются либо одинарные, либо двойные кавычки. Поэтому при объединении JavaScript и HTML-кода есть смысл придерживаться одного «стиля» кавычек для JavaScript, а другого – для HTML.

В следующем примере строка «Спасибо» в JavaScript-выражении заключена в одинарные кавычки, а само выражение, в свою очередь, заключено в двойные кавычки как значение HTML-атрибута обработчика событий:

```
<a href="" onclick="alert('Спасибо')">Щелкни меня</a>
```

Чтобы включить в строковый литерал специальные символы, необходимо использовать нотацию escape-последовательностей:

Код	Вывод	Код	Вывод
\0	нулевой символ (символ NUL)	\v	вертикальная табуляция
\'	одинарная кавычка	\t	табуляция
\"	двойная кавычка	\b	забой
\\	обратный слэш	\f	подача страницы
\n	новая строка	\uXXXX	кодовая точка Юникода
\r	возврат каретки	\xXX	символ из кодировки Latin-1

Строки полезны для хранения данных, которые можно представить в текстовой форме. Некоторые из наиболее частых операций со строками – это проверка их длины, построение строки с помощью операций строковой конкатенации + и +=, проверка на существование или местоположение подстрок с помощью метода `indexOf()`, либо извлечение подстрок с помощью метода `substring()`.

Объект `String` используется, чтобы представить и конструировать последовательность символов.

Значением **свойства `length`** объекта **`String`** является количество символов в строке. Для пустой строки это значение равно нулю.

Синтаксис: **`объект.length`**

Пример:

```
var txt = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";
var sln = txt.length; // 33
```

Позиции символов строки нумеруются от нуля до **`объект.length-1`**.

Стандартные методы объекта String

Метод charAt возвращает символ, находящийся в данной позиции строки.

Синтаксис: `объект.charAt(позиция)`

Аргументы: `позиция` - любое числовое выражение.

Результат: строковое значение.

Метод `charAt` возвращает строку, состоящую из символа, расположенного в данной позиции примитивного значения строкового объекта. Если позиция лежит вне этого диапазона, то возвращается пустая строка.

Пример:

```
"Строка".charAt(0) // символ С
```

Метод charCodeAt возвращает код символа, находящегося в данной позиции строки.

Синтаксис: `объект.charCodeAt(позиция)`

Аргументы: позиция - любое числовое выражение.

Результат: числовое значение.

Метод `charAt` возвращает число, равную коду `Unicode` символа, расположенного в данной позиции примитивного значения строкового объекта. Если позиция лежит вне этого диапазона, то возвращается `NaN`.

Пример:

```
"Строка".charCodeAt(0) // шестнадцатеричный код буквы "С": 421
```

Метод concat возвращает конкатенацию строк.

Синтаксис: `объект.concat(строка0, строка1, …, строкаN)`

Аргументы: `строка0`, `строка1`, …, `строкаN` - любые строковые выражения.

Результат: строковое значение.

Метод `concat` возвращает новую строку, являющуюся конкатенацией исходной строки и аргументов метода. Этот метод эквивалентен операции

```
объект + строка0 + строка1 + … + строкаN
```

Пример:

```
"Мороз и солнце. ".concat("День чудесный.")
```

Метод fromCharCode создает строку из символов, заданных кодами Unicode.

Синтаксис: `String.fromCharCode(код1, код2, …, кодN)`

Аргументы: `код1`, `код2`, …, `кодN` - числовые выражения.

Результат: строковое значение.

Метод `fromCharCode` создает новую строку (но не строковый объект), которая является конкатенацией символов Unicode с кодами `код1`, `код2`, …, `кодN`. Это статический метод объекта String, поэтому для доступа к нему не нужно специально создавать строковый объект.

Пример:

```
var s = String.fromCharCode(65, 66, 67); // s равно "ABC"
```

Метод indexOf возвращает позицию первого вхождения заданной подстроки.

Синтаксис: `объект.indexOf(подстрока [,начало]?)`

Аргументы: `подстрока` - любое строковое выражение, `начало` - любое числовое выражение.

Результат: числовое значение

Метод `indexOf` возвращает первую позицию подстроки в примитивном значении строкового объекта. Если задан необязательный аргумент `начало`, то поиск ведется, начиная с позиции `начало`; если нет, то с позиции 0, т. е. с первого символа строки. Если начало отрицательно, то оно принимается равным нулю; если начало больше, чем `объект.length-1`, то оно принимается равным `объект.length-1`. Если объект не содержит данной подстроки, то возвращается значение `-1`.

Поиск ведется слева направо. В остальном этом метод идентичен методу `lastIndexOf`.

Примеры:

1)

```
var str = "Пожалуйста, найдите, где происходит 'locate'";
var pos = str.indexOf("locate");
```

2)

```
//подсчет количества вхождений подстроки pattern в строку str
function occur(str, pattern) {
  var pos = str.indexOf(pattern);
  for (var count = 0; pos != -1; count++)
```

```
        pos = str.indexOf(pattern, pos + pattern.length);
      return count;
    }
```

Метод lastIndexOf возвращает позицию последнего вхождения заданной подстроки.

Синтаксис: `объект.lastIndexOf(подстрока [,начало]?)`

Аргументы: `подстрока` - любое строковое выражение, `начало` - любое числовое выражение.

Результат: числовое значение.

Метод `lastIndexOf` возвращает последнюю позицию подстроки в примитивном значении строкового объекта. Если задан необязательный аргумент начало, то поиск ведется, начиная с позиции `начало`; если нет, то с позиции 0, т. е. с первого символа строки. Если `начало` отрицательно, то оно принимается равным нулю; если `начало` больше, чем `объект.length-1`, то оно принимается равным `объект.length-1`. Если объект не содержит данной подстроки, то возвращается значение -1.

Поиск ведется справа налево. В остальном этом метод идентичен методу `indexOf`.

Примеры:

1)
```
var n = "Белый кит".lastIndexOf("кит"); // n равно 6
```
2)
```
var str = "Пожалуйста, найдите, где происходит 'locate'";
var pos = str.lastIndexOf("locate");
```

Метод localeCompare сравнивает две строки с учетом языка операционной системы.

Синтаксис: `объект.localeCompare(строка1)`

Аргументы: `строка1` - любое строковое выражение

Результат: число

Метод `localeCompare` сравнивает две строки с учетом национальных установок операционной системы. Он возвращает -1, если примитивное значение объекта меньше `строки1`, +1, если оно больше `строки1`, и 0, если эти значения совпадают.

Метод match сопоставляет строку с регулярным выражением.

Синтаксис: `объект.match(регвыр)`

Аргументы: `регвыр` - любое регулярное выражение.

Результат: массив строк.

Метод `match` сопоставляет регулярное выражение `регвыр` с примитивным значением строкового объекта. Результатом сопоставления является массив найденных подстрок или `null`, если соответствий нет. При этом:

- Если `регвыр` не содержит опцию глобального поиска, то выполняется метод `регвыр.exec(объект)` и возвращается его результат. Результирующий массив содержит в элементе с индексом 0 найденную подстроку, а в остальных элементах — подстроки, соответствующие подвыражениям `регвыр`, заключенным в круглые скобки.
- Если `регвыр` содержит опцию глобального поиска, то метод `регвыр.exec(объект)` выполняется до тех пор, пока находятся соответствия. Если n — количество найденных соответствий, то результатом является массив из n элементов, которые содержат найденные подстроки. Свойству `регвыр.lastIndex` присваивается номер позиции в исходной строке, указывающий на первый символ после последнего найденного соответствия, или 0, если соответствий не найдено.

Следует помнить, что метод регвыр.exec изменяет свойства объекта регвыр.

Примеры:

```
var src = "Он сказал: <I>Я ухожу</I> и добавил: <I>До свидания</I>.";
var res = src.match(/<i>.*?<\/i>/i);  // Я ухожу
var res = src.match(/<i>.*?<\/i>/ig); // Я ухожу,До свидания
```

Метод replace сопоставляет строку с регулярным выражением и заменяет найденную подстроку новой подстрокой.

Синтаксис: объект.replace(регвыр,строка)
объект.replace(регвыр,функция)

Аргументы: регвыр - регулярное выражение, строка - строковое выражение, функция - имя функции или декларация функции.

Результат: новая строка.

Метод replace сопоставляет регулярное выражение регвыр с примитивным значением строкового объекта и заменяет найденные подстроки другими подстроками. Результатом является новая строка, которая является копией исходной строки с проведенными заменами. Способ замены определяется опцией глобального поиска в регвыр и типом второго аргумента.

- Если регвыр не содержит опцию глобального поиска, то выполняется поиск первой подстроки, соответствующей регвыр и производится ее замена. Если регвыр содержит опцию глобального поиска, то выполняется поиск всех подстрок, соответствующих регвыр, и производится их замена.
- Если вторым аргументом является строка, то замена каждой найденной подстроки производится на нее. При этом строка может содержать такие свойства объекта RegExp, как $1, …, $9, lastMatch, lastParen, leftContext и rightContext.

Пример:

```
"Вкусные яблоки, сочные яблоки.".replace(/яблоки/g, "груши"))
// Вкусные груши, сочные груши.
```

Если вторым аргументом является функция, то замена каждой найденной подстроки производится вызовом этой функции. Функция имеет следующие аргументы. Первый аргумент - это найденная подстрока, затем следуют аргументы, соответствующие всем подвыражениям регвыр, заключенным в круглые скобки, предпоследний аргумент - это позиция найденной подстроки в исходной строке, считая с нуля, и последний аргумент - это сама исходная строка.

Пример, показывающий, как с помощью метода replace можно написать функцию преобразования градусов Фаренгейта в градусы Цельсия:

```
function myfunc($0,$1) {
  return (($1-32) * 5 / 9) + "C";
}
function f2c(x) {
  var s = String(x);
  return s.replace(/(\d+(\.\d*)?)F\b/, myfunc);
}
document.write(f2c("212F")); // выведет строку 100C
```

Следует помнить, что этот метод изменяет свойства объекта регвыр.

Пример использования replace для замены всех вхождений подстроки в строку:

```
var str = "foobarfoobar";
str=str.replace(/foo/g,"xxx"); // str = "xxxbarxxxbar"
```

Метод search ищет сопоставление строки с регулярным выражением.

Синтаксис: `объект.search(регвыр)`

Аргументы: `регвыр` - любое регулярное выражение.

Результат: числовое выражение.

Метод `search` сопоставляет регулярное выражение `регвыр` с примитивным значением строкового объекта. Результатом сопоставления является позиция первой найденной подстроки, считая с нуля, или -1, если соответствий нет. При этом опция глобального поиска в `регвыр` игнорируется, и свойства `регвыр` не изменяются.

Примеры:

```
var src = "Он сказал: <I>Я ухожу</I> и добавил: <I>До свидания</I>.";
var n = src.search(/<i>.*?<\/i>/i);   // n равно 11
```

Метод slice извлекает часть строки и возвращает новую строку.

Синтаксис: `объект.slice(начало [,конец]?)`

Аргументы: `начало` и `конец` - любые числовые выражения.

Результат: новая строка.

Метод `slice` возвращает подстроку примитивного значения строкового объекта, от позиции начало до позиции конец, не включая ее. Если конец не задан, то возвращается подстрока, начиная с позиции начало и до конца исходной строки.

Если значение `начало` отрицательно, то оно заменяется на `объект.length+начало`. Если значение `конец` отрицательно, то оно заменяется на `объект.length+конец`. Иными словами, отрицательные аргументы трактуются как смещения от конца строки.

Примеры:

1)

```
var str = "Яблоко, Банан, Киви";
var res = str.slice(7, 13); // "Банан"
```

2)

```
document.write("ABCDEF".slice(2,-1))  // выведет строку CDE.
```

Метод split разбивает строку на массив подстрок.

Синтаксис: `объект.split(разделитель [,число]?)`

Аргументы: `разделитель` - строковое или регулярное выражение, `число` - числовое выражение.

Результат: массив строк (объект `Array`).

Метод `split` разбивает примитивное значение объекта на массив подстрок и возвращает его. Разбиение на подстроки производится следующим образом. Исходная строка просматривается слева направо в поисках разделителя. Как только он найден, подстрока от конца предыдущего разделителя (или от начала строки, если это первое вхождение разделителя) до начала найденного добавляется в массив подстрок. Таким образом, сам разделитель в текст подстроки не попадает.

Необязательный аргумент `число` задает максимально возможный размер результирующего массива. Если он задан, то после выделения числа подстрок метод завершает работу, даже если просмотр исходной строки не закончен.

Разделитель может быть задан либо строкой, либо регулярным выражением. Существует несколько случаев, требующих особого рассмотрения:

- Если разделитель не задан, то результирующий массив состоит из одного элемента, равного исходной строке.
- Если разделитель — пустая строка или регулярное выражение, соответствующее пустой строке, то результирующий массив состоит из `объект.length` элементов, каждый из которых содержит один символ исходной строки.
- Если разделитель — регулярное выражение, то очередная подстрока выделяется вся-

кий раз, когда нашлось соответствие регулярному выражению, и соответствующая подстрока служит разделителем.

Примеры:

1)

```
var s="a1b2c3d".split(/\d/); // массив ["a","b","c","d"]
```

2) регулярное выражение используется для задания тегов HTML в качестве разделителя

```
document.write("Текст <B>полужирный</B> и
<EM>курсивный</EM>".split(/<\/?([^<>]+)>/));
// выведет строку Текст ,полужирный, и ,курсивный.
```

Метод substr возвращает подстроку, заданную позицией и длиной.

Синтаксис: `объект.substr(позиция [,длина]?)`

Аргументы: `позиция` и `длина` - числовые выражения.

Результат: строковое значение

Метод `substr` возвращает подстроку примитивного значения строкового объекта, начинающуюся с данной позиции и содержащую `длина` символов. Если `длина` не задана, то возвращается подстрока, начиная с данной позиции и до конца исходной строки. Если `длина` отрицательна или равна нулю, то возвращается пустая строка.

Если позиция больше или равна `объект.length`, то возвращается пустая строка. Если позиция отрицательна, то она трактуется как смещение от конца строки, т. е. заменяется на `объект.length+позиция`.

Примеры:

```
var src = "abcdef";
var s1 = src.substr(1, 3);    // "bcd"
var s2 = src.substr(1);       // "bcdef"
var s3 = src.substr(-1);      // "f"
```

Метод substring возвращает подстроку, заданную начальной и конечной позициями.

Синтаксис: `объект.substring(начало [,конец])`

Аргументы: `начало` и `конец` - числовые выражения.

Результат: строковое значение

Метод `substring` возвращает подстроку примитивного значения строкового объекта, от позиции начало до позиции конец, не включая ее. Если конец не задан, то возвращается подстрока, начиная с позиции начало и до конца исходной строки.

Отрицательные аргументы или равные `NaN` заменяются на нуль; если аргумент больше длины исходной строки, то он заменяется на нее. Если начало больше конца, то они меняются местами. Если начало равно концу, то возвращается пустая строка.

Примеры:

1)

```
var src = "abcdef";
var s1 = src.substring(1, 3);      // "bc"
var s2 = src.substring(1, -1);     // "a"
var s3 = src.substring(-1, 1);     // "a"
```

2)

```
var str = "Яблоко, Банан, Киви";
var res = str.substring(7, 13);    // "Банан"
```

Метод toLowerCase преобразует все буквы строки в строчные.

Синтаксис: `объект.toLowerCase()`

Результат: новая строка.

Метод `toLowerCase` возвращает новую строку, в которой все буквы исходной строки заменены на строчные. Остальные символы исходной строки не изменяются. Исходная строка остается прежней.

Пример:

```
document.write("объект String".toLowerCase()) // объект string
```

Метод toUpperCase преобразует все буквы строки в прописные.

Синтаксис: `объект.toUpperCase()`

Результат: новая строка

Метод `toUpperCase` возвращает новую строку, в которой все буквы исходной строки заменены на прописные. Остальные символы исходной строки не изменяются. Исходная строка остается прежней.

Пример:

```
document.write("объект String".toUpperCase()) // ОБЪЕКТ STRING
```

Метод toString преобразует объект в строку.

Синтаксис: `объект.toString()`

Результат: строковое значение

Метод `toString` возвращает примитивное значение строкового объекта.

Примеры:

```
var number = -507;
string_value = number.toString();
var n = 17;
binary_string = n.toString(2); // Вернет "10001"
octal_string = "0" + n.toString(8); // Вернет "021"
hex_string = "0x" + n.toString(16); // Вернет "0x11"
```

Метод valueOf возвращает примитивное значение объекта.

Синтаксис: `объект.valueOf()`

Результат: строковое значение

Метод `valueOf` возвращает примитивное значение строкового объекта.

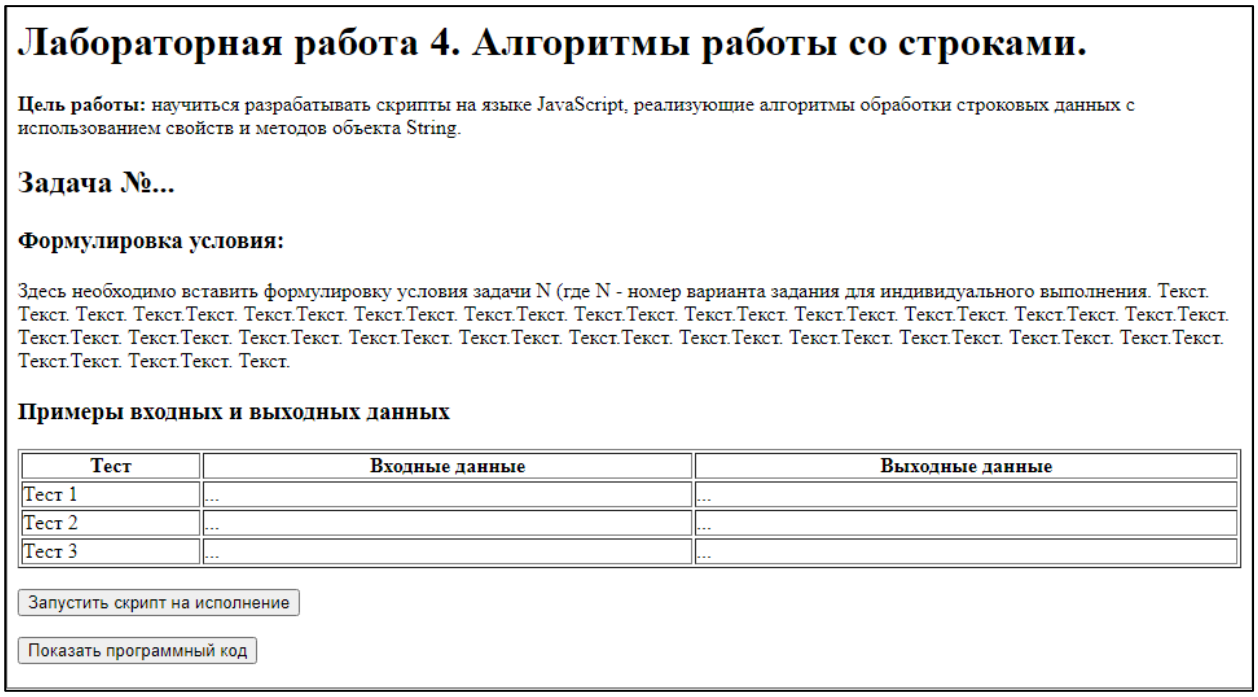
Пример:

```
var x = new String('Привет, мир');
document.write (x.valueOf()); // Отобразит 'Привет, мир'
```

Задания для выполнения

Задание 1. Разработайте скрипт, реализующий алгоритм обработки строковых данных. Ввод исходных данных осуществлять с использованием функции `promt`. Вывод полученных результатов осуществлять с использованием функции `alert`. Сохраните скрипт в файле **script_4.js**.

Задание 2. Создайте по образцу html-документ **lab_4.html** и подключите к нему внешний скрипт **script_4.js**. Для тестирования работоспособности скрипта разработайте 3 теста (включающих наборы входных и выходных данных). Выполните тестирование и отладку скрипта в браузере.



Лабораторная работа 4. Алгоритмы работы со строками.

Цель работы: научиться разрабатывать скрипты на языке JavaScript, реализующие алгоритмы обработки строковых данных с использованием свойств и методов объекта String.

Задача №...

Формулировка условия:

Здесь необходимо вставить формулировку условия задачи N (где N - номер варианта задания для индивидуального выполнения. Текст. Текст. Текст. Текст.

Примеры входных и выходных данных

Тест	Входные данные	Выходные данные
Тест 1	...	...
Тест 2	...	...
Тест 3	...	...

Запустить скрипт на исполнение

Показать программный код

Образец html-документа

Варианты индивидуальных задач к Заданию 1:

1. Даны две строки s_1 и s_2. Посчитать, сколько раз в строке s_2 встречается первый символ строки s_1.
2. Дана строка s, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Найти номер самого длинного слова. Если таких слов несколько, то указать все их номера.
3. Дана строка s. Известно, что среди символов данной строки имеется хотя бы один пробел и что первый символ строки s отличен от пробела. Преобразовать строку s, заменив все малые буквы латинского алфавита, предшествующие первому пробелу, одноименными большими.
4. Даны две строки s_1 и s_2, имеющие одинаковую длину. Образовать строку s, в которой должны чередоваться символы строк s_1 и s_2.
5. Дана строка s, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Если количество слов в строке s четно, то поменять местами ее средние слова. В противном случае на место среднего слова строки s поместить пробелы.
6. Дана строка s. Преобразовать ее, заменив в ней каждую точку многоточием (то есть тремя точками), а все восклицательные знаки вопросительными.
7. Дана строка s. Преобразовать эту строку так, чтобы ее символы шли в обратном порядке по отношению к их исходному расположению.
8. Дана строка s, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить, сколько слов состоит из одинаковых символов, например: аааaa, eee.
9. Определить долю пробелов в данной строке s.

10. Даны две строки s_1 и s_2, имеющие различную длину. Посчитать в них количество попарно одинаковых символов.

11. Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить число слов, в которых последняя буква в слове не повторяется.

12. Дана строка *s*. Определить, является ли она ”палиндромом”. Если нет, то выяснить, станет ли строка *s* ”палиндромом” после удаления из нее всех пробелов.

13. Дана строка s_1. Образовать строку s_2, включив в нее символы строки s_1, расположенные на нечетных позициях, кроме пробелов и знаков препинания.

14. Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить, повторяется ли в тексте первое слово.

15. Дана строка *s*. Вывести ее на экран, подчеркивая (ставя минусы в соответствующих позициях следующей строки) все входящие в него цифры.

16. Дана строка s. Исключить из нее все лишние пробелы, то есть вместо каждой группы пробелов оставить только один. При исключении пробелов текст сдвигается влево.

17. Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Найти номер самого короткого слова. Если таких слов несколько, то указать все их номера.

18. Даны две строки s_1 и s_2, имеющие различную длину. Посчитать в них количество попарно различных символов.

19. Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить, сколько раз стоящие рядом два слова начинаются на одну и ту же букву.

20. Дана строка *s*. Определить в ней долю символов, являющихся буквами латинского алфавита.

21. Дана строка *s*, состоящая из букв латинского алфавита. Верно ли, что в данной строке гласные буквы (*a, e, i, o, u*) чередуются с согласными?

22. Дана строка *s*. Известно, что среди символов данной строки есть по крайней мере один восклицательный знак и что первый символ этой строки отличен от восклицательного знака. Выяснить, имеется ли среди символов строки *s*, предшествующих первому восклицательному знаку, пара соседствующих одинаковых символов.

23. Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Если первое и последнее слова строки имеют одинаковую длину, то поменять их местами. В противном случае оставить строку *s* без изменения.

24. Дана строка *s*. Удалить из нее все группы букв вида *abcd*.

25. Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить число слов в строке, состоящих из нечетного числа символов.

26. Дана строка *s*, состоящая из букв латинского алфавита. Преобразовать данную строку, упорядочив в ней буквы по алфавиту.

27. Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить, содержит ли данная строка слова, заканчивающиеся символом * .
28. Дана строка *s*. Известно, что среди символов данной строки имеется хотя бы один пробел и что первый символ строки *s* отличен от пробела. Преобразовать строку *s*, удалив из нее все символы, предшествующие первому пробелу и не являющиеся буквами латинского алфавита.
29. Дана строка *s*. Выяснить, имеются ли такие натуральные *i* и *j*, что $i < j$ и *i*-тый символ строки *s* совпадает и (*i*+1)-ым символом, а *j*-тый — с (*j*+1)-ым символом строки *s*.
30. Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Преобразовать строку s, удалив из нее все повторные вхождения слов.

Лабораторная работа 5. Массивы с числовыми индексами

Цель работы: научиться разрабатывать скрипты на языке JavaScript, реализующие алгоритмы обработки одномерных и многомерных (двумерных) числовых массивов.

Теоретический материал по теме

Массив – тип данных, содержащий (хранящий) пронумерованные значения. Каждое пронумерованное значение называется **элементом массива**, а число, с которым связывается элемент, называется его **индексом**.

Так как JavaScript – нетипизированный язык, элемент массива может иметь любой тип, причем разные элементы одного массива могут иметь разные типы. Элементы массива могут даже содержать другие массивы, что позволяет создавать массивы массивов. Индексация элементов массива начинается с 0.

Массивы в JavaScript являются динамическими: они могут увеличиваться и уменьшаться в размерах по мере необходимости; нет необходимости объявлять фиксированные размеры массивов при их создании или повторно распределять память при изменении их размеров.

1. Создание массивов

Создать массив с помощью **литерала**, который представляет собой простой список разделенных запятыми элементов массива в квадратных скобках. Значения в литерале массива не обязательно должны быть константами - это могут быть любые выражения, в том числе и литералы объектов.

Примеры:

```
let empty = []; // Пустой массив
let primes = [2, 3, 5, 7, 11]; // Массив с 5 числовыми элементами
let misc = [1.1, true, "a"]; // 3 элемента разных типов
let base = 1024;
let table = [base, base+1, base+2, base+3];
let b = [[1,{x:1, y:2}], [2, {x:3, y:4}]];
let count = [1,,3]; // Массив из 3 элементов, средний элемент не
определен.
let undefs = [,,]; // Массив из 2 элементов, оба не определены.
```

Другой способ создания массива состоит в вызове конструктора `Array()`. Вызвать конструктор можно тремя разными способами:

1) Вызвать конструктор без аргументов:

```
let arr = new Array();
```
В этом случае будет создан пустой массив, эквивалентный литералу [].

2) Вызвать конструктор с единственным числовым аргументом, определяющим длину массива:

```
let arr = new Array(10);
```
В этом случае будет создан пустой массив указанной длины. Такая форма вызова конструктора `Array()` может использоваться для предварительного распределения памяти под массив, если заранее известно количество его элементов. При этом в массиве не сохраняется никаких значений.

3) Явно указать в вызове конструктора значения первых двух или более элементов массива или один нечисловой элемент:

```
let arr = new Array(5, 4, 3, 2, 1, "тест");
```
В этом случае аргументы конструктора становятся значениями элементов нового массива.

Примеры:

```
let a = new Array( ); // пустой массив
let a = new Array(5, 4, 3, 2, 1, "testing, testing");
let a = new Array(10); // массив из 10 элементов
```

2. Чтение и запись элементов массива

Доступ к элементам массива осуществляется с помощью оператора []. Слева от скобок должна присутствовать ссылка на массив. Внутри скобок должно находиться произвольное выражение, возвращающее неотрицательное целое значение. Этот синтаксис пригоден как для чтения, так и для записи значения элемента массива.

Примеры:

```
value = a[0];
a[1] = 3.14;
i = 2;
a[i] = 3;
a[i + 1] = "hello";
a[a[i]] = a[0];
my['salary'] *= 2;
a[1.23] = true; //создается новое свойство с именем "-1.23”
```

3. Длина массива

Все массивы имеют свойство `length`, устанавливающее количество элементов в массиве. Поскольку массивы могут иметь неопределенные элементы, свойство `length` всегда на единицу больше, чем самый большой номер элемента массива.

Примеры:

```
let a = new Array(); //a.length == 0 (ни один элемент не определен)
a = new Array(10); //a.length == 10 (определены пустые элементы 0-9)
a = new Array(1,2,3); //a.length == 3 (определены элементы 0-2)
a = [4, 5]; // a.length == 2 (определены элементы 0 и 1)
a[5] = 1; // a.length == 6 (определены элементы 0, 1 и 5)
a[49] = 0; // a.length == 50 (определены элементы 0, 1, 5 и 49)
```

Свойство length массива доступно как для чтения, так и для записи. Если установить свойство length в значение, меньшее текущего, массив укорачивается до новой длины; любые элементы, не попадающие в новый диапазон индексов, отбрасываются, и их значения теряются. Если сделать свойство length большим, чем его текущее значение, в конец массива добавляются новые неопределенные элементы, увеличивая массив до нового размера.

4. Обход элементов массива

Одним из самых старых способов перебора элементов массива является цикл for по цифровым индексам.

Примеры:

1)
```
let fruits = ["манго", "банан", "вишня", "персик"];
for(let i = 0; i < fruits.length; i++)
   alert(fruits[i]);
```
2)
```
let lookup_table = new Array(1024);
for(let i = 0; i < lookup_table.length; i++)
   lookup_table[i] = i * 512;
```
3)
```
let k = Number(prompt('Введите размер массива ', '5'));
let x = new Array(k);
for (let i=0; i<=k-1; i++)
     x[i] = Math.round((Math.random()*100));
```

Также для массивов возможен и другой вариант цикла, `for..of`. Цикл `for..of` не предоставляет доступа к номеру текущего элемента, только к его значению.

Пример:
```
let fruits = ["Яблоко", "Апельсин", "Слива"];
// проходит по значениям
for (let fruit of fruits) {
  alert( fruit );
}
```

Методы класса Array

Метод join()

Метод `Array.join()` преобразует все элементы массива в строки, объединяет их и возвращает получившуюся строку. В необязательном аргументе методу можно передать строку, которая будет использоваться для отделения элементов в строке результата. Если строка-разделитель не указана, используется запятая.

Пример:
```
let arr = [1,2,3];
arr.join();              // '1,2,3'
arr.join("-");           // '1-2-3'
```

Метод reverse()

Метод `Array.reverse()` меняет порядок следования элементов в массиве на обратный и возвращает переупорядоченный массив. Перестановка выполняется непосредственно в исходном массиве, т.е. этот метод не создает новый массив с переупорядоченными элементами, а переупорядочивает их в уже существующем массиве.

Пример:
```
let arr = [1,2,3];
arr.reverse().join();         // "3,2,1"
```

Метод sort()

Метод `Array.sort()` сортирует элементы в исходном массиве и возвращает отсортированный массив. Если метод `sort()` вызывается без аргументов, сортировка выполняется в алфавитном порядке (для сравнения элементы временно преобразуются в строки, если это необходимо). Неопределенные элементы переносятся в конец массива.

Для сортировки в каком-либо ином порядке, отличном от алфавитного, методу `sort()` можно передать функцию сравнения в качестве аргумента. Эта функция устанавливает, какой из двух ее аргументов должен следовать раньше в отсортированном списке.

Если первый аргумент должен предшествовать второму, функция сравнения должна возвращать отрицательное число. Если первый аргумент должен следовать за вторым в отсортированном массиве, то функция должна возвращать число больше нуля. А если два значения эквивалентны (т.е. порядок их следования не важен), функция сравнения должна возвращать 0.

Пример:

```
let arr = [33, 4, 1111, 222];
arr.sort();              // Алфавитный порядок: 1111, 222, 33, 4
arr.sort(function(a,b) {
     // Числовой порядок: 4, 33, 222, 1111
     return a-b;    // Возвращает значение < 0, 0 или > 0
                    // в зависимости от порядка сортировки a и b
});
// Сортируем в обратном направлении, от большего к меньшему
arr.sort(function(a,b) {return b-a});
```

Метод concat()

Метод `Array.concat()` создает и возвращает новый массив, содержащий элементы исходного массива, для которого был вызван метод `concat()`, и значения всех аргументов, переданных методу `concat()`. Если какой-либо из этих аргументов сам является массивом, его элементы добавляются в возвращаемый массив. Следует, однако, отметить, что рекурсивного превращения массива из массивов в одномерный массив не происходит. Метод concat() не изменяет исходный массив.

Примеры:

```
let arr = [1,2,3];
arr.concat(4, 5);        // Вернет [1,2,3,4,5]
arr.concat([4,5]);       // Вернет [1,2,3,4,5]
arr.concat([4,5],[6,7])  // Вернет [1,2,3,4,5,6,7]
arr.concat(4, [5,[6,7]]) // Вернет [1,2,3,4,5,[6,7]]
```

Метод slice()

Метод `Array.slice()` возвращает фрагмент, или подмассив, указанного массива. Два аргумента метода определяют начало и конец возвращаемого фрагмента. Возвращаемый массив содержит элемент, номер которого указан в первом аргументе, плюс все последующие элементы, вплоть до (но не включая) элемента, номер которого указан во втором аргументе.

Если указан только один аргумент, возвращаемый массив содержит все элементы от начальной позиции до конца массива. Если какой-либо из аргументов имеет отрицательное значение, он определяет номер элемента относительно конца массива. Так, аргументу -1 соответствует последний элемент массива, а аргументу -3 - третий элемент массива с конца.

Примеры:

```
let arr = [1,2,3,4,5];
arr.slice(0,3);          // Вернет [1,2,3]
arr.slice(3);       // Вернет [4,5]
arr.slice(1,-1);    // Вернет [2,3,4]
arr.slice(-3,-2);   // Вернет [3]
```

Метод splice()

Метод `Array.splice()` - это универсальный метод, выполняющий вставку или удаление элементов массива. В отличие от методов `slice()` и `concat()`, метод `splice()` изменяет исходный массив, относительно которого он был вызван. Обратите

внимание, что методы splice() и slice() имеют очень похожие имена, но выполняют совершенно разные операции.

Метод splice() может удалять элементы из массива, вставлять новые элементы или выполнять обе операции одновременно. Элементы массива при необходимости смещаются, чтобы после вставки или удаления образовывалась непрерывная последовательность.

Первый аргумент метода splice() определяет позицию в массиве, начиная с которой будет выполняться вставка и/или удаление. Второй аргумент определяет количество элементов, которые должны быть удалены (вырезаны) из массива. Если второй аргумент опущен, удаляются все элементы массива от указанного до конца массива. Метод splice() возвращает массив удаленных элементов или (если ни один из элементов не был удален) пустой массив.

Первые два аргумента метода splice() определяют элементы массива, подлежащие удалению. За этими аргументами может следовать любое количество дополнительных аргументов, определяющих элементы, которые будут вставлены в массив, начиная с позиции, указанной в первом аргументе.

Примеры:

```
let arr = [1,2,3,4,5,6,7,8];
arr.splice(4);           // Вернет [5,6,7,8], arr = [1,2,3,4]
arr.splice(1,2);         // Вернет [2,3], arr = [1,4]
arr.splice(1,1);         // Вернет [4]; arr = [1]
arr = [1,2,3,4,5];
arr.splice(2,0,'a','b'); // Вернет []; arr = [1,2,'a','b',3,4,5]
```

Методы push() и pop()

Методы push() и pop() позволяют работать с массивами как со стеками. Метод push() добавляет один или несколько новых элементов в конец массива и возвращает его новую длину. Метод pop() выполняет обратную операцию - удаляет последний элемент массива, уменьшает длину массива и возвращает удаленное им значение. Обратите внимание, что оба эти метода изменяют исходный массив, а не создают его модифицированную копию.

Примеры:

```
var arr = [];       // Создать пустой массив
arr.push('zero');   // Добавить значение в конец
arr.push('one',2);  // Добавить еще два значения
```

Методы unshift() и shift()

Методы unshift() и shift() ведут себя почти так же, как push() и pop(), за исключением того, что они вставляют и удаляют элементы в начале массива, а не в конце. Метод unshift() смещает существующие элементы в сторону больших индексов для освобождения места, добавляет элемент или элементы в начало массива и возвращает новую длину массива. Метод shift() удаляет и возвращает первый элемент массива, смещая все последующие элементы на одну позицию вниз, чтобы занять место, освободившееся в начале массива.

Многомерные массивы

Согласно практике решения множества задач и в случае работы со сложными наборами данных возникает необходимость построения многомерных массивов данных. Фактически, Javascript не поддерживает подобных массивов, но синтаксис языка подразумевает использование в качестве элементов массива любого объекта, в том числе и других массивов.

Пример:

```
// Создать многомерный массив
let table = new Array(10);          // В таблице 10 строк
```

```
for(let i = 0; i < table.length; i++)
     table[i] = new Array(10);     // В каждой строке 10 столбцов
// Заполнить массив значениями и вывести в консоль
for(let row = 0, str = ''; row < table.length; row++) {
     for(let col = 0; col < table[row].length; col++) {
          table[row][col] = (row+1)*(col+1);
          str += table[row][col] + '  ';
     }
     console.log(str + '\n');
     str = '';
}
```

Задания для выполнения

Задание 1. Разработайте скрипты, реализующие алгоритм обработки одномерного числового массива. Ввод исходных данных в массив осуществлять двумя способами (на выбор): 1) с использованием функции `promt`; 2) с использованием функции `Math.random()`, генерирующей псевдослучайные числа. Вывод полученных результатов осуществлять двумя способами (на выбор): 1) с использованием функции `alert`;) в ячейки таблицы html-документа. Сохраните скрипт в файле **script_5_1.js**.

Задание 2. Разработайте скрипты, реализующий алгоритм обработки двумерного числового массива. Ввод исходных данных в массив осуществлять двумя способами (на выбор): 1) с использованием функции `promt`; 2) с использованием функции `Math.random()`, генерирующей псевдослучайные числа. Вывод полученных результатов осуществлять двумя способами (на выбор): 1) с использованием функции `alert`;) в ячейки таблицы html-документа. Сохраните скрипт в файле **script_5_2.js**.

Задание 3. Создайте по образцу html-документы **lab_5_1.html** и **lab_5_2.html** и подключите к ним внешние скрипты **script_5_1.js** и **script_5_2.js** соответственно. Для тестирования работоспособности каждого скрипта разработайте по 3 теста (включающих наборы входных и выходных данных). Выполните тестирование и отладку скриптов в браузере.

Лабораторная работа 5. Массивы с числовыми индексами.

Цель работы: научиться разрабатывать скрипты на языке JavaScript, реализующие алгоритмы обработки одномерных и многомерных (двумерных) числовых массивов.

Задача №...

Формулировка условия:

Здесь необходимо вставить формулировку условия задачи N (где N - номер варианта задания для индивидуального выполнения. Текст. Текст. Текст. Текст.

Примеры входных и выходных данных

Тест	Входные данные	Выходные данные
Тест 1	...	...
Тест 2	...	...
Тест 3	...	...

[Запустить скрипт на исполнение]

[Показать программный код]

Образец html-документа

Варианты индивидуальных задач к Заданию 1:

1. Дан одномерный числовой массив. Если он упорядочен по убыванию, то вывести его на экран в обратном порядка; в противном случае вывести на экран номер первого элемента, нарушающего упорядоченность.
2. Дан одномерный числовой массив. Все элементы массива, кратные 3, домножить на их индекс, а затем найти среднее арифметическое элементов преобразованного массива.
3. В одномерном числовом массиве поменять местами элементы с четными и нечетными индексами, если оба они кратны минимальному, и заменить эти элементы их индексами в обратном случае.
4. Дан одномерный числовой массив, все элементы которого различны. Если сумма всех элементов, расположенных между максимальным и минимальным элементами массива, меньше 10, то вычислить произведение ненулевых элементов одномерного числового массива; в противном случае найти количество элементов массива, кратных минимальному, и их произведение.
5. Дан одномерный числовой массив, все элементы которого различны. Подсчитать в нем количество четных элементов, расположенных между наибольшим и наименьшим элементами массива.
6. Дан одномерный числовой массив, все элементы которого различны. Найти наименьшее положительное значение элемента массива и его номер. Если этот номер окажется больше 5, то заменить все элементы, расположенные между наибольшим и наименьшим элементами массива, их квадратами, иначе — кубами.
7. Дан одномерный числовой массив, все элементы которого различны. Посчитать в нем количество соседств из двух положительных чисел. Если это количество кратно 3, то вывести на экран все элементы расположенные до максимального, иначе — после максимального.
8. Дан одномерный числовой массив, все элементы которого различны. Посчитать сумму всех неотрицательных элементов массива с четными индексами. Если эта сумма окажется больше 7, то заменить наибольший и наименьший элементы массива их произведением, иначе — поменять их местами.
9. Дан упорядоченный массив $a_1,\ a_2,\ \ldots a_n$. Известно, что число x принадлежит отрезку числовой оси, вмещающему заданный массив. Определить номер k, для которого $a_{k-1} < x < a_k$. Если k окажется кратным 3, то вывести на экран элементы массива в обратном порядке.
10. Дан одномерный числовой массив, все элементы которого различны. Посчитать сумму кубов элементов с нечетными индексами, расположенных до наибольшего элемента массива, и произведение индексов всех элементов, которые по числовому значению превосходят минимальный, но не превосходят максимальный элемент массива.
11. Дан одномерный числовой массив. Определить, сколько в нем имеется пар совпадающих по величине соседних чисел; вывести на экран их индексы.
12. Дан одномерный числовой массив. Добавить к каждому элементу массива, кроме первого, значение предыдущего. После такого преобразования вычислить среднее арифметического наибольшего и наименьшего элементов массива.
13. Найти наибольший и наименьший элементы одномерного числового массива, а также подсчитать количество элементов, кратных наименьшему элементу массива.

14. Дан одномерный числовой массив, все элементы которого различны. Заменить в нем четные элементы, расположенные после максимального, их квадратами, а нечетные элементы, расположенные после максимального — их кубами.

15. Вычислить произведение всех ненулевых элементов одномерного числового массива с нечетными индексами. Заменить все элементы массива, равные нулю, отношением их индекса к значению максимального элемента.

16. Заменить в одномерном числовом массиве элементы, большие числа M, на число a, введенное с клавиатуры, а меньшие M — отношением $\frac{a}{i}$, где i — индекс элемента.

17. В одномерном числовом массиве заменить отрицательные элементы их квадратами, а положительные разделить на числовое значение минимального элемента.

18. Дан одномерный числовой массив. Определить количество элементов этого массива, которые имеют четные индексы и являются кратными 3, а также вычислить сумму и произведение элементов, кратных либо первому, либо последнему элементам массива.

19. Определить в одномерном числовом массиве число соседств из двух чисел, сумма которых равна 5. Вычислить произведение всех элементов, удовлетворяющих этому условию.

20. Найти суммы и произведения отрицательных и положительных элементов одномерного числового массива. Подсчитать, сколько имеется в массиве элементов, равных нулю.

21. Дан одномерный числовой массив. Домножить на 3 все его положительные элементы с нечетными индексами; все отрицательные элементы, имеющие четные индексы, уменьшить в 5 раз.

22. Дан одномерный числовой массив. Вычислить сумму квадратов его элементов с четными индексами, а также произведение всех ненулевых элементов с нечетными индексами.

23. Определить в одномерном числовом массиве количество соседств из взаимно противоположных и не взаимно обратных чисел.

24. Добавить к каждому нечетному элементу одномерного целочисленного массива его номер, а затем все четные элементы заменить произведением номера элемента на отношение наибольшего элемента массива к наименьшему.

25. Найти наибольший элемент одномерного числового массива из стоящих на нечетных местах и наименьший из стоящих на четных местах. Если сумма этих элементов больше значения первого элемента массива, то уменьшить их в 2 раза, иначе — увеличить каждый из них на 3.

26. Дан одномерный числовой массив, все элементы которого различны. Посчитать, сколько в нем содержится элементов, совпадающих по абсолютной величине с номером. Если таких элементов нет, то вычислить сумму и произведение наибольшего и наименьшего элементов массива.

27. В одномерном числовом массиве найти количество и сумму элементов, кратных 3, а также среднее арифметического всех элементов, имеющих четные индексы и кратных 2.

28. Определить в одномерном числовом массиве количество соседств из чисел разного знака. Вывести на экран последовательность из чисел одного знака, имеющую наибольшую длину.

29. Дан одномерный числовой массив. Найти сумму корней квадратных из абсолютных величин его элементов, а также произведение всех отрицательных элементов с четными номерами и произведение всех положительных элементов с нечетными номерами.

30. Посчитать в одномерном числовом массиве количество элементов, равных первому отрицательному. Вывести на экран самую длинную подпоследовательность из элементов, равных нулю.

Варианты индивидуальных задач к Заданию 2:

1. Даны натуральное число n и действительные числа $a_1,\ a_2,\ ...a_n, b_1,\ b_2,\ ...b_n$. Получить действительную матрицу $\left[c_{ij}\right]$, для которой $c_{ij} = \dfrac{a_i + b_j}{\sqrt{|a_i| + |b_j|}}$, где $i, j = 1, 2, ..., n$.

2. Дана действительная квадратная матрица порядка n (n — натуральное число). Заменить нулями все элементы матрицы, находящиеся на ее главной и побочной диагоналях, кроме наибольшего и наименьшего из них.

3. Дана действительная матрица размера $M \times N$. Найти сумму наибольших значений элементов ее строк, а также произведение наименьших элементов ее столбцов.

4. Дано целое число n. Построить целочисленную квадратную матрицу порядка n вида

$$\begin{pmatrix} n & n-1 & n-2 & \dots & 1 \\ 0 & n & n-1 & \dots & 2 \\ 0 & 0 & n & \dots & 3 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & n \end{pmatrix}.$$

5. Дана целочисленная квадратная матрица порядка 8. Найти наименьшее из значений элементов столбца, который обладает наибольшей суммой модулей элементов. Если таких столбцов несколько, то взять первый из них.

6. Дана действительная квадратная матрица порядка n, в которой не все элементы равны нулю. Получить новую матрицу путем деления всех элементов данной матрицы, лежащих ниже главной диагонали, на ее наибольший по модулю элемент.

7. Дана действительная квадратная матрица порядка n. Вычислить сумму тех из ее элементов, расположенных на главной диагонали и выше нее, которые превосходят по величине все элементы, расположенные ниже главной диагонали. Если на главной диагонали и выше нее нет элементов с указанным свойством, то вывести сообщение об этом.

8. Дана целочисленная квадратная матрица порядка n. Заменить нулями все неотрицательные элементы этой матрицы, находящиеся на ее побочной диагонали.

9. Дана действительная квадратная матрица порядка n. Переменной y присвоить значение, равное скалярному произведению строки и столбца (рассматриваемых как векторы), на пересечении которых находится наименьший элемент матрицы (в предположении, что такой элемент единственный).

10. Даны натуральные числа n, m, действительная матрица размера $m \times n$. Найти среднее арифметическое каждого из столбцов, имеющих четные номера.
11. Даны натуральное число n, действительная квадратная матрица порядка n, действительные $a_1, \ldots, a_{n+5}$. Элементы последовательности $a_1, \ldots, a_{n+5}$ домножить на 3, если наибольший элемент матрицы (в предположении, что такой элемент единственный) расположен на главной диагонали, и на 0.5 в противном случае.
12. В данной действительной матрице размера $m \times n$ поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащий элемент с наименьшим значением. Предполагается, что эти элементы единственны.
13. Дано натуральное число n. Получить действительную матрицу $\left[a_{ij}\right]$ ($i, j = 1, 2, ..., n$), для которой $a_{ij} = \dfrac{i \cdot j}{i + j}$.
14. Дана целочисленная квадратная матрица порядка n. Переменной y присвоить значение наибольшего из элементов матрицы, расположенных на главной диагонали и ниже ее.
15. Дано натуральное число n. Выяснить, сколько положительных элементов содержит матрица $\left[a_{ij}\right]$ ($i, j = 1, 2, ..., n$), если $a_{ij} = \sin\left(i + \dfrac{j}{2}\right)$.
16. Дана действительная квадратная матрица порядка n. Рассмотрим те ее элементы, которые расположены в строках, начинающихся с отрицательного элемента. Найти суммы тех из них, которые находятся соответственно ниже, выше и на побочной диагонали.
17. Дано натуральное число n. Получить действительную матрицу $\left[a_{ij}\right]$ ($i, j = 1, 2, ..., n$), первая строка которой задается формулой $a_{1j} = 2j + 3$ ($j = 1, 2, ..., n$), вторая строка задается формулой $a_{2j} = j - \dfrac{3}{2 + \dfrac{1}{j}}$ ($j = 1, 2, ..., n$), а каждая следующая строка есть сумма двух предыдущих.
18. Дана действительная квадратная матрица порядка n. В строках с отрицательным элементом на главной диагонали найти наибольший из всех элементов.
19. Дана действительная матрица размера $m \times n$, все элементы которой различны. В каждой строке выбирается элемент с наименьшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы этого элемента.
20. Дана действительная матрица размера $m \times n$. Найти среднее арифметическое наибольшего и наименьшего элементов этой матрицы из лежащих на ее побочной диагонали.
21. В данной действительной квадратной матрице порядка n найти сумму элементов строки, в которой расположен элемент с наименьшим значением. Предполагается, что такой элемент единственный.
22. Дана действительная квадратная матрица порядка n. Найти сумму элементов матрицы, расположенных в строках с отрицательным элементом на главной диагонали.
23. Дан двумерный целочисленный массив размером $m \times n$. Некоторый элемент этого массива назовем ”седловой точкой”, если он является одновременно наименьшим в своей строке и наибольшим в своем столбце. Напечатать номера строки и столбца какой-нибудь ”седловой точки” и напечатать число 0, если такой точки нет.

24. Дана действительная квадратная матрица порядка *n*. Получить целочисленную квадратную матрицу того же порядка, в которой элемент равен единице, если соответствующий ему элемент исходной матрицы больше элемента, расположенного в его строке на главной диагонали, и равен нулю в противном случае.
25. Даны натуральное число *m* и целочисленная квадратная матрица порядка *m*. Строку с номером *i* матрицы назовем отмеченной, если $a_i > 0$, и неотмеченной в противном случае. Посчитать число отрицательных элементов, расположенных в отмеченных строках.
26. Дана действительная квадратная матрица порядка *n*. Построить последовательность действительных чисел a_1, a_2, $\ldots a_n$ по правилу: если в *i*-той строке матрицы элемент, принадлежащий главной диагонали, отрицателен, то a_i равно сумме элементов *i*-той строки, предшествующих первому отрицательному элементу; в противном случае a_i равно сумме последних элементов *i*-той строки, начиная с первого по порядку неотрицательного элемента.
27. Дано натуральное число *n*. Получить действительную матрицу $\left[c_{ij}\right]$, для которой $c_{ij} = \dfrac{i+j}{i^3+j^3}$, где *i, j* = 1, 2, ..., *n*.
28. Дана действительная квадратная матрица порядка *n*. В строках с отрицательным элементом на побочной диагонали найти сумму всех элементов.
29. Дан двумерный целочисленный массив размером *m*×*n*. Известно, что среди его элементов два и только два равны между собой. Напечатать их индексы.
30. Дано натуральное число *n*. Выяснить, сколько положительных элементов содержит матрица $\left[a_{ij}\right]$ (*i, j* = 1, 2, ..., *n*), если $a_{ij} = \sin\left(\dfrac{i^2 - j^2}{n}\right)$.

Лабораторная работа 6. Объекты как ассоциативные массивы

Цель работы: научиться разрабатывать скрипты на языке JavaScript, реализующие алгоритмы обработки объектов - ассоциативных массивов.

Теоретический материал по теме

Объекты – это составной тип данных, они объединяют множество значений в единый модуль и позволяют сохранять и извлекать значения по их именам.

Объекты – это неупорядоченные коллекции **свойств**, каждое из которых имеет свои **имя** (**ключ**) и **значение**. Ключи свойств должны быть строками или символами (обычно строками). Значения могут быть любого типа.

Под **ассоциативным массивом** подразумевают массив, в котором в качестве ключей применяются строки. То есть речь идёт о совокупности пар «ключ-значение». Таким образом, в ассоциативном массиве любое значение связано с конкретным ключом, а доступ к этому значению производится по имени ключа.

Основное отличие простого индексного массива от ассоциативного в том, что в обычном массиве значения хранятся под номерами, которые называются **индексами**, а в ассоциативном - под именами, которые называются **ключами**.

Создание объектов при помощи литералов

Литерал объекта – это выражение, которое создает и инициализирует новый объект

всякий раз, когда производится вычисление этого выражения. С помощью единственного **литерала объекта** можно создать множество новых объектов, если этот литерал поместить в тело цикла или функции, которая будет вызываться многократно.

Литерал объекта – это заключенный в фигурные скобки список свойств (пар «имя–значение»), разделенных запятыми.

Ассоциативный массив объявляется так же, как и обычный. Пишется слово `let`, далее прописывается его имя, знак равно, затем, обратите внимание, вместо квадратных пишутся фигурные скобки. В фигурных скобках через запятую располагаются элементы, которые состоят из ключа и значения.

Примеры:

```
let empty = {}; // Объект без свойств
let point = { x:0, y:0 };
let circle = { x:point.x, y:point.y+1, radius:2 };
let homer = {
  "name": "Homer Simpson",
  "age": 34,
  "married": true,
  "occupation": "plant operator",
  'email': "homer@example.com"
};
```

Для удобного восприятия и чтения кода оптимальный вариант формировать массив, где каждый элемент прописан на отдельной строке, а также заключать ключи в кавычки. Имя свойства из нескольких слов должно обязательно заключаться в кавычки.

Доступ к значениям свойств объектов

Для доступа к значениям свойств объекта используется оператор **«точка»**:

```
obj.property
```

Пример:

```
let book = new Object();
book.title = "Программирование на JavaScript "
book.chapter1 = new Object();
book.chapter1.title = "Введение в JavaScript";
book.chapter1.pages = 11;
book.chapter2 = { title: "Лексическая структура", pages: 6 };
alert("Заголовок: " + book.title + "\n\t" +
"Глава 1 " + book.chapter1.title + "\n\t" +
"Глава 2 " + book.chapter2.title);
```

Чтобы получить доступ к свойству, можно также использовать и квадратные скобки: `obj["property"]`.

Квадратные скобки позволяют взять ключ из переменной, например: `obj[varWithKey]`.

Пример:

```
let user = {
  name: "John",
  age: 30,
  "likes birds": true
}
// присваивание значения свойству
user["likes birds"] = true;
// получение значения свойства
alert(user["likes birds"]); // true
let key = "likes birds";
```

```
// то же самое, что и user["likes birds"] = true;
user[key] = true;
```

Перечисление свойств объекта

```
function DisplayPropertyNames(obj) {
var names = "";
   for(var name in obj) names += name + "\n";
   alert(names);
}
```

Проверка существования свойств

```
// Если объект o имеет свойство с именем "x", установить его
if ("x" in o) o.x = 1;
// Если свойство x существует и его значение не равно undefined,
установить его.
if (o.x !== undefined) o.x = 1;

o.x = undefined; // свойство x будет существовать, но не будет
иметь значения

// Если свойство doSomething существует и не содержит значение
null
// или undefined, тогда предположить, что это функция и ее сле-
дует вызвать!
if (o.doSomething) o.doSomething();
```

Удаление свойств

```
delete book.chapter2;
// удаление свойства
delete user["likes birds"];
```

При удалении свойства его значение не просто устанавливается в значение undefined; оператор delete действительно удаляет свойство из объекта.

Цикл for/in демонстрирует это отличие: он перечисляет свойства, которым было присвоено значение undefined, но не перечисляет удаленные свойства.

Доступ к свойствам объекта возможен также при помощи оператора [], который обычно применяется при работе с массивами:

```
object.property
object["property"]
```

При обращении к свойству объекта с помощью нотации массивов [] имя свойства задается в виде строки, которая может создаваться и изменяться во время работы программы.

Пример 1. Код, в котором читаются и объединяются в одну строку свойства address0, address1, address2 и address3 объекта customer:

```
var addr = "";
for(i = 0; i < 4; i++) {
   addr += customer["address" + i] + '\n';
}
```

Пример 2: Программа, обращающаяся к сетевым ресурсам для вычисления текущей суммы инвестиций пользователя на фондовом рынке. Программа разрешает пользователю вводить названия любых имеющихся у него акций, а также количество каждого вида акций. Т.к. пользователь вводит названия акций во время исполнения программы, нет способа узнать имена свойств заранее. Поэтому доступ к свойствам объекта portfolio при помощи оператора «точка» невозможен, необходимо использовать оператор []:

```
var stock_name = get_stock_name_from_user();
var shares = get_number_of_shares();
portfolio[stock_name] = shares;
var value = 0;
for (stock in portfolio) {
   // Для каждого вида акций в портфеле получаем стоимость
   // одной акции и умножаем ее на число акций.
   value += get_share_value(stock) * portfolio[stock];
}
```

Задания для выполнения

Задание 1. Разработайте скрипты, реализующие алгоритм решения задачи с использованием ассоциативных массивов. Ввод исходных данных осуществлять с использованием формы <form> html-документа. Вывод полученных результатов осуществлять с использованием функции `alert`. Сохраните скрипты в файле **script_6.js**.

Задание 2. Создайте по образцу html-документ **lab_6.html** и подключите к нему внешние скрипты **script_6.js**. Для тестирования работоспособности скриптов разработайте по 3 теста (включающих наборы входных и выходных данных). Выполните тестирование и отладку скриптов в браузере.

Лабораторная работа 6. Объекты как ассоциативные массивы.

Цель работы: научиться разрабатывать скрипты на языке JavaScript, реализующие алгоритмы обработки объектов - ассоциативных массивов.

Задача №...

Формулировка условия:

Здесь необходимо вставить формулировку условия задачи N (где N - номер варианта задания для индивидуального выполнения. Текст. Текст. Текст. Текст.

Примеры входных и выходных данных

Тест	Входные данные	Выходные данные
Тест 1	...	...
Тест 2	...	...
Тест 3	...	...

Форма ввода данных:

Имя:
Фамилия:
Пол: ○ Мужской ○ Женский
☐ Даю согласие на обработку персональных данных
Ввести данные | Очистить форму | Получить результаты

Образец html-документа

Варианты индивидуальных задач к Заданию 1:

1. Дан ассоциативный массив, содержащий сведения об учениках школы: имя и фамилия ученика, а также название класса (года обучения и буквы), в котором он учится. Выяснить, имеются ли в школе однофамильцы.

2. Дан ассоциативный массив, содержащий сведения о нескольких автомобилях: марка автомобиля, его номер и фамилия владельца. Найти фамилии владельцев и номера автомобилей данной марки.

3. Дан ассоциативный массив, в котором содержится информация о военнослужащих некоторого войскового подразделения: фамилия, имя, отчество, возраст, рост (от 140 до 210 см). Проверить, имеется ли в данном подразделении хотя бы два человека одного роста.

4. Дан ассоциативный массив, содержащий сведения об экспортируемых товарах: указывается наименование товара, страна, импортирующая товар, и объем поставляемой партии в штуках. Найти страны, в которые экспортируется данный товар, и общий объем его экспорта.

5. Дан ассоциативный массив, который содержит номера телефонов сотрудников учреждения: указывается фамилия сотрудника, его инициалы и номер телефона. Найти телефон сотрудника по его фамилии и инициалам.

6. Дан ассоциативный массив, содержащий информацию о багаже нескольких пассажиров: фамилия пассажира, количество вещей в его багаже и общий вес вещей. Найти пассажира, в багаже которого средний вес одной вещи отличается не более чем на 0.3 кг от общего среднего веса вещи.

7. Дан ассоциативный массив, содержащий сведения о книгах. Сведения о каждой из книг — это фамилия автора, название и год издания. Найти названия книг данного автора, изданных с 1960 года.

8. Дан ассоциативный массив, содержащий сведения о веществах: указывается название вещества, его удельный вес и проводимость (проводник, полупроводник, диэлектрик). Найти удельные веса и названия всех полупроводников.

9. Дан ассоциативный массив, содержащий сведения об игрушках: указывается название игрушки (например, кукла, кубики, мяч, конструктор), ее стоимость в рублях и возрастные границы детей, для которых игрушка предназначена (например, для детей от двух до пяти лет). Вывести на экран названия наиболее дорогих игрушек, цена которых отличается от цены самой дорогой игрушки не более чем на 1 руб.

10. Дан ассоциативный массив, в котором содержится информация о студентах некоторого вуза: указывается фамилия, имя, отчество студента, его пол (мужской, женский), возраст (от 16 до 35), а также номер курса, на котором он учится (от 1 до 5). Узнать номер курса, на котором обучается наибольший процент мужчин.

11. Дан ассоциативный массив, содержащий сведения о кубиках: размер каждого кубика (длина ребра в сантиметрах) и его цвет (красный, желтый, синий или зеленый). Найти количество кубиков каждого из перечисленных цветов и их суммарный объем.

12. Дан ассоциативный массив, содержащий сведения об учениках класса: имя и фамилия ученика, а также отметки, полученные учениками в последней четверти по физике, математике и литературе. Найти учеников, которые по всем предметам имеют отметки не ниже четырех баллов.

13. Дан ассоциативный массив, содержащий сведения о различных датах. Каждая дата — это число, месяц, год. Найти все весенние даты и записать их в файл g.

14. Дан ассоциативный массив, содержащий сведения о клиентах банка: фамилия, имя, отчество клиента, номер его счета и сумма вклада на этом счете. Найти всех клиентов, имеющих несколько счетов, и вычислить общую сумму вкладов каждого из них.

15. Дан ассоциативный массив, содержащий различные даты. Каждая дата — это число, месяц, год. Найти самую позднюю дату.

16. Дан ассоциативный массив, содержащий сведения о почтовых посылках: фамилия отправителя, фамилия получателя, вес посылки в килограммах и дата ее отправления (число, месяц, год). Найти всех получателей, на имя которых в течение последнего месяца пришло несколько посылок, и вычислить общий вес посылок, полученных каждым из них.

17. Дан ассоциативный массив, содержащий информацию о подписчиках периодических изданий: фамилия подписчика, его домашний адрес, число выписываемых газет и число выписываемых журналов. Найти всех подписчиков, которые выписывают не менее трех газет и двух журналов, и указать их адреса.

18. Дан ассоциативный массив, в котором содержится информация о владельцах дачных участков садоводческого товарищества: фамилия владельца, номер дачного участка и его площадь (в кв. м.). Найти всех владельцев, имеющих несколько дачных участков, указать для каждого владельца номера всех его дачных участков, а также их общую площадь.

19. Дан ассоциативный массив, содержащий сведения о читателях библиотеки: фамилия, имя, отчество человека, год, с которого он является читателем библиотеки, а также номер его читательского билета. Вывести на экран информацию о самом ”старом” читателе библиотеке.

20. Дан ассоциативный массив, содержащий сведения о сотрудниках фирмы: фамилия, имя, отчество сотрудника, его возраст и стаж работы в фирме (в годах). Найти самых молодых сотрудников фирмы (имеющих наименьший возраст), стаж работы которых составляет не менее 3 лет.

21. Дан ассоциативный массив, содержащий различные даты: месяц (январь, февраль, март и т. д.) и число (от 1 до 31). Переписать из этого файла все летние даты в файл summer, а все зимнее даты — в файл winter.

22. Дан ассоциативный массив, содержащий сведения об учебниках для средней школы: указываются предмет (например, ”Физика”), автор учебника и номер класса, для которого он предназначен (от 1 до 11). Найти предмет, по которому существует наибольшее количество учебников различных авторов для данного класса.

23. Дан ассоциативный массив, в котором содержатся сведения о служащих учреждения: фамилия и инициалы, пол (мужской, женский) и семейное положение. Выяснить, имеется ли вероятность того, что некоторые из сотрудников учреждения состоят в семейном браке.

24. Дан ассоциативный массив, содержащий информацию об абонентах операторов сотовой связи за последний месяц: указываются оператор, номер телефона абонента, число звонков внутри сети и их общая продолжительность. Найти всех абонентов, имевших наибольшее число звонков внутри сети при их наименьшей средней продолжительности.

25. Дан ассоциативный массив, содержащий сведения о вакансиях рабочих мест: указываются требуемая профессия, ежемесячный размер оплаты труда и номер телефона, по которому можно связаться с работодателем. Найти информацию о самых высокооплачиваемых рабочих местах по данной профессии (где предлагаемый ежемесячный размер оплаты труда выше среднего по данной профессии).

26. Дан ассоциативный массив, содержащий информацию о жителях Кемеровской области: фамилия человека, город, в котором он живет и его адрес (название улицы, номер дома и номер квартиры). Найти фамилии двух любых жителей Кемеровской области, живущих в разных городах по одинаковому адресу.
27. Дан ассоциативный массив, в котором содержатся сведения о студентах некоторого вуза: фамилия, имя, отчество, пол, возраст, курс, причем в фамилии, имени и отчестве — не более 12 букв, пол указывается буквами М и Ж, возраст — целое от 16 до 35, курс — целое от 1 до 5. Выяснить, на каком курсе наибольший процент мужчин.
28. Дан ассоциативный массив, содержащий сведения о пассажирах авиакомпании: фамилия, имя, отчество пассажира, номер рейса. Выяснить, имеются ли на данном рейсе однофамильцы, если да, то указать их ФИО.
29. Дан ассоциативный массив, содержащий информацию о имеющихся на складе комплектующих для компьютеров: указывается наименование детали, ее стоимость в рублях и количество таких деталей. От покупателя поступил заказ на приобретение партии комплектующих, в котором указаны их наименования и требуемое число деталей каждого наименования. Выяснить, имеется ли на складе достаточное число комплектующих для выполнения данного заказа. В случае, если заказ может быть выполнен, подсчитать его полную стоимость.
30. Дан ассоциативный массив, в котором содержатся сведения о служащих учреждения: фамилия, имя, отчество, пол (мужской, женский), а также стаж работы в данном учреждении. Найти самые распространенные среди сотрудников учреждения мужское и женское имена.

Список рекомендуемой литературы

Основная учебная литература

1. Брокшмидт, К. Пользовательский интерфейс приложений для Windows 8, созданных с использованием HTML, CSS и JavaScript : учебный курс / К. Брокшмидт. – 2-е изд., исправ. – Москва : Национальный Открытый Университет «ИНТУИТ», 2016. – 396 с. - URL: http://biblioclub.ru/index.php?page=book&id=429247 (дата обращения: 26.12.2019).
2. Хоган, Б. Книга веб-программиста. Секреты профессиональной разработки веб-сайтов =WebDevelopmentRecipes / Б. Хоган. - Санкт-Петербург [и др.] : Питер, 2013. - 288 с. - ISBN 978-5-459-01510-2. - Текст : непосредственный.

Дополнительная учебная литература

1. Богданов, М.Р. Разработка клиентских приложений Web-сайтов: курс / М.Р. Богданов. – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2010. – 228 с. : ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=233745 (дата обращения: 23.10.2020). – Текст : электронный. —Режим доступа: для авториз. пользователей.
2. Громов, Ю. Ю. Основы Web-инжиниринга: разработка клиентских приложений : учебное пособие / Ю. Ю. Громов, О. Г. Иванова, С. В. Данилкин ; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Тамбовский государственный технический университет». – Тамбов : Издательство ФГБОУ ВПО «ТГТУ», 2012. – 240 с. – URL: http://biblioclub.ru/index.php?page=book&id=277648 (дата обращения: 26.12.2019). – Текст : электронный. — Режим доступа: для авториз. пользователей.
3. Кингсли, Х. Э. JavaScript в примерах : учебное пособие / Х. Э. Кингсли, Х. К. Кингсли. — Москва : ДМК Пресс, 2009. — 272 с. — ISBN 978-5-94074-668-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/1271 (дата обращения: 24.10.2020). — Режим доступа: для авториз. пользователей.
4. Хэррон, Д. Node.js. Разработка серверных веб-приложений в JavaScript / Д. Хэррон ; перевод с английского А. А. Слинкина. — Москва : ДМК Пресс, 2012. — 144 с. — ISBN 978-5-94074-809-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/50571 (дата обращения: 24.10.2020). — Режим доступа: для авториз. пользователей.
5. Штефен, В. Разработка приложений для Windows 8 с помощью HTML5 и JavaScript. Подробное руководство : руководство / В. Штефен ; перевод с английского А. А. Слинкин. — Москва : ДМК Пресс, 2013. — 344 с. — ISBN 978-5-94074-921-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/58696 (дата обращения: 24.10.2020). — Режим доступа: для авториз. пользователей.