Лабораторная работа JS №1

ИСПОЛЬЗОВАНИЕ БАЗОВЫХ КОНСТРУКЦИЙ ЯЗЫКА JAVASCRIPT

Цель работы: изучить базовые конструкции языка JavaScript

Этапы выполнения:

1. *Изучить теоретические материалы*
2. *Реализовать примеры*
3. *Выполнить общие задания*
4. *Выполнить индивидуальные задания. Вариант определяется следующим образом: 1-1, 2-2, 3-3, 4-4, 5-1, 6-2 и т.д.*
5. *Если работа будет сдана не в срок, необходимо выполнить дополнительные задания.*

Теоретические материалы:

Общий обзор языка.

Основные определения Любая программа оперирует некими данными: именем стилевого класса, размерами элемента, цветом шрифта и прочие. JavaScript может манипулировать данными, относящимися к разным типам. Тип данных описывает их возможные значения и набор применимых к ним операций. Типы данных бывают простыми и сложными. Сущность, относящаяся к простому типу данных, может хранить только одно значение (это строковые, числовые и логические типы данных). Сущность сложного типа данных может хранить сразу несколько значений. Например – массивы. Другой пример сложного типа данных – объекты. Для создания механизма управления страницами на клиентской стороне было предложено использовать объектную модель документа. Суть модели в том, что каждый HTML-контейнер — это объект, который характеризуется тройкой:

- Свойства
- Методы
- События

Объекты – это сложные сущности, позволяющие хранить сразу несколько значений разных типов данных, они представляют собой блоки, из которых строится JavaScript. Применяются для возвращения значений и изменения состояния форм, страниц, браузера и определенных программистом переменных. Объекты можно сопоставить с существительными. Кошка, автомобиль, дом, компьютер, форма – все это существительные, они могут быть представлены как объекты.

2. Экземпляры объекта – сущности, хранящие реальные данные и созданные на основе этого объекта. То есть конкретный, реально существующий дом, находящийся по заданному адресу можно рассматривать, как экземпляр объекта типа дом.

3. Свойства – набор внутренних параметров объекта. Используются для того, чтобы различать экземпляры одного объекта – например, все экземпляры типа дом. Свойства сравнимы с прилагательными и ссылаются на уникальные для каждого экземпляра объекта особенности. Один и тот же объект может обладать многими свойствами: дом может быть большим и маленьким, синим

и красным. Разные объекты могут обладать одинаковыми свойствами: дерево, так же, как и дом, может быть большим и маленьким, синим и красным... Большинство свойств объекта мы можем изменять, воздействуя на них через методы.

4. Методы – это действие или способ, при помощи которого мы можем изменять определенные свойства объекта, то есть управлять этими объектами, а также в некоторых случаях менять их содержимое.

5. События – это очень важное в программировании на JavaScript понятие. События главным образом порождаются пользователем, являются следствиями его действий. Если пользователь нажимает кнопку мыши, то происходит событие, которое называется Click.

Если экранный указатель мыши движется по ссылке HTML-документа, происходит событие MouseOver. Существует несколько различных событий.

6. Оператор – это команда, инструкция для компьютера. Встретив в программе тот или иной оператор, машина четко его выполняет.

7. Функция – это определенная последовательность операторов, то есть набор команд, последовательное выполнение которых приводит к какому-то результату. Например, выполнение кем-то заданной Вами функции (процедуры) "возьми стакан, открой кран, набери в него воды и принеси мне" приведет к результату: Вы получите стакан воды из-под крана.

8. Переменная – в языках программирования переменные используются для хранения данных определенного типа, например параметров свойств объекта. Каждая переменная имеет свое имя (идентификатор) и хранит только одно значение, которое может меняться в ходе выполнения программы. Данные могут быть разных типов: целое число, десятичная дробь, логическая константа, текстовая строка.

Понятие объектной модели применительно к JavaScript

При загрузке HTML-страницы в браузер интерпретатор языка создает объекты со свойствами, определенными значениями тэгов страницы. Для правильного использования объектных моделей следует четко понимать, как браузер компонует страницы и, тем самым, создает иерархию объектов. При загрузке страницы просматриваются сверху вниз, тем самым последовательно происходит компоновка страницы и ее отображение в окне браузера. А это означает, что и объектная модель страницы также формируется последовательно, по мере ее обработки. Поэтому невозможно обратится из сценария, расположенного ранее какой- либо формы на странице, к элементам этой формы. Всегда следует помнить о том, что браузер последовательно сверху вниз интерпретирует содержимое HTML-страницы.

Еще один аспект работы с объектами языков сценариев заключается в том, что нельзя изменить свойства объектов. Браузер обрабатывает страницу только один раз, компонуя и отображая ее. Поэтому попытка в сценарии изменить свойство отображенного элемента страницы, обречена на провал. Только повторная загрузка страницы приведет к желаемому результату.

Размещение операторов языка JavaScript на странице

Встроить сценарий JavaScript в HTML-страницу можно несколькими способами.

1. Задание операторов языка внутри тэга <script> языка HTML.

Для внедрения в HTML-страницу сценария JavaScript в спецификацию языка HTML был введен тэг-контейнер <script>...</script>, внутри которого могут располагаться операторы языка JavaScript. Обычно браузеры, не поддерживающие какие-нибудь тэги HTML, просто их игнорируют, анализируя, однако, содержимое пропускаемых тэгов с точки зрения синтаксиса языка HTML, что может приводить к ошибкам при отображении страницы. Во избежание подобной ситуации следует помещать операторы языка JavaScript в контейнер комментария

<! -- ... //--
>, как показано ниже
<script (language="javascript")>
<!--
операторы javascript
//-->
</script>

Параметр language задает используемый язык сценариев. В случае языка JavaScript его значение задавать не обязательно, так как этот язык используется браузерами по умолчанию.

Примечание:

символы // перед закрывающим тэгом комментария --> являются оператором комментария JavaScript. Он необходим для правильной работы интерпретатора.

Документ может содержать несколько тэгов <script>, расположенных в любом месте документа. Все они последовательно обрабатываются интерпретатором JavaScript по мере отображения частей документа в окне браузера. В связи с этим ссылка на переменную, определенную в сценарии, размещенном в конце документа, может привести к генерации ошибки интерпретатора при обращении к такой переменной из сценария в начале документа.

2. Задание файла с кодом JavaScript.

Тэг <script> имеет параметр src, позволяющий связать встраиваемый сценарий с внешним файлом, содержащим программный код на языке JavaScript. В качестве значения параметра задается полный или относительный URL-адрес ресурса. Задание закрывающего тэга </script> обязательно, независимо от того, заданы или нет операторы внутри тэга.

Следующий фрагмент кода связывает документ HTML с файлом- источником, содержащим некоторый набор функций:

<script language="JavaScript" src="http://url/file.js">
операторы javascript
</script>

Примечание:

связываемый внешний файл не должен содержать тэгов HTML и должен иметь расширение .js.

3. Использование выражений JavaScript в качестве значений параметров тэгов HTML.

Переменные и выражения JavaScript можно использовать в качестве значений параметров тэгов HTML.

Например:

```
<a href="javascript: window.open('name.htm', '_self')">
<img src="images/ch1.png" border="0">
</a>
```

4. Определение обработчика событий в тэге HTML.

1.2. Язык ядра JavaScript

Синтаксис языка

Язык JavaScript чувствителен к регистру.

Приложение JavaScript представляет собой набор операторов языка (команд), последовательно обрабатываемых встроенным в браузер интерпретатором. Каждый оператор можно располагать в отдельной строке. В этом случае разделитель ‘;’, отделяющий один оператор от другого, не обязателен. Его используют только в случае задания нескольких операторов на одной строке. Любой оператор можно расположить в нескольких строках без всякого символа продолжения.

Например, следующие два вызова функции alert эквивалентны:

```
...
alert("Подсказка");
alert(
"Подсказка"
);
...
```

Нельзя перемещать на другую строку единый строковый литерал – он должен располагаться полностью на одной строке текста программы или разбит на два строковых литерала, соединенных операцией конкатенации ‘+’:

```
...
alert("Подсказка"); // правильно
alert("Под
сказка"); // неправильно
alert("Под" +
"сказка"); // правильно (но браузер выведет текст одной строкой!)
...
```

Пробельные символы в тексте программы являются незначащими, если только они не используются в строковых литералах.

В JavaScript строковые литералы можно задавать двумя равноправными способами – последовательность символов, заключенная в двойные или одинарные кавычки:

```
"Анна"

'Анна'
```

В строковых литералах можно использовать ESC-последовательности, которые начинаются с символа обратной наклонной черты, за которой следует обычный символ. Некоторые подобные комбинации трактуются как один специальный символ.

Таблица 1.

\b	Возврат на один символ
\f	Переход на новую страницу
\n	Переход на новую строку
\r	Возврат каретки
\t	Горизонтальная табуляция Ctrl-I
\'	Апостроф
\"	Двойные кавычки
\\	Обратная наклонная черта

ESC-последовательности форматирования используются при отображении информации в диалоговых окнах, отображаемых функциями alert(), prompt() и confirm(), а также, если методом document.write()

записывается содержимое элемента pre.

Комментарии в программе JavaScript двух видов: однострочные и многострочные:

// комментарий, расположенный на одной строке.

/*

комментарий, расположенный на нескольких строках.

*/

Ссылка на объект осуществляется по имени, заданному параметром name тэга HTML, с использованием точечной нотации. Например, пусть в документе задана форма с двумя полями ввода:

<form name="form1">

Фамилия: <input type = "text" name = "student" size = 20>

Курс: <input type = "text" name = "course" size = 2>

</form>

Для получения фамилии студента, введенного в первом поле ввода, в программе JavaScript следует использовать ссылку document.form.student.value, а чтобы определить курс, на котором обучается студент, необходимо использовать ссылку document.form.course.value.

Переменные и литералы в JavaScript

В JavaScript все переменные вводятся с помощью одного ключевого слова var. Синтаксическая конструкция для ввода в программе новой переменной с именем name1 выглядит следующим образом:

var name1;

Объявленная таким образом переменная name1 имеет значение 'undefined' до тех пор, пока ей не будет присвоено какое-либо другое значение, которое можно присвоить и при ее объявлении:

var name1 = 5;

var name1 = "новая строковая переменная";

JavaScript поддерживает четыре простых типа данных:

• Целый
• Вещественный
• Строковый
• Логический (булевый)

Для присваивания переменным значений основных типов применяются литералы – буквальные значения данных соответствующих типов.

Выражения JavaScript

Выражение – комбинация переменных, литералов и операторов, в результате вычисления которой получается одно единственное значение. Переменные в выражениях должны быть инициализированы.

1. Присваивание

Оператор присваивания (=) рассматривается как выражение присваивания, которое вычисляется равным выражению правой части, и в то же время он присваивает вычисленное значение выражения переменной, заданной в левой части:

var name2=10;

2. Арифметическое выражение Вычисляемым значением арифметического выражения является число. Создаются с помощью арифметических операторов.

Таблица 2.

Оператор	**Действие**
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Остаток от деления целых чисел
++	Увеличение значения на единицу

3. Логическое выражение

Вычисляемым значением логического выражения может быть true или false. Для создания используются операторы сравнения или логические операторы, применяемые к переменным любого типа.

Таблица 3.

Операторы сравнения	Значение	Логические Операторы	Значение
= =	Равно	&&	логическое И
! =	Не равно	||	логическое ИЛИ
>=	Больше или равно	!	логическое НЕ
<=	Меньше или равно		
>	Строго больше		
<	Строго меньше		

4. Строковые выражения Вычисляемым значением строкового выражения является число. В JavaScript существует только один строковый оператор – оператор конкатенации (сложения) строк:

string1 = “Моя ” + “строка”

1.3. Управляющие конструкции языка JavaScript

Операторы JavaScript

Операторы служат для управления потоком команд в JavaScript. Блоки операторов должны быть заключены в фигурные скобки.

1. Операторы выбора

• условный оператор if

Эта управляющая структура используется, когда необходимо выполнить некий программный код в зависимости от определенных условий. Также предусмотрена конструкция if-else (если-тогда-иначе).

```
if (условие_1)
{
оператор_1; // эти операторы выполняются, если условие_1
верно
оператор_2;
}
else
{
оператор_3; // эти операторы выполняются, если условие_1 ложно
оператор_4;
}
```

Условие для проверки (вопрос компьютеру) записывается сразу после слова if в круглых скобках. После этого в фигурных скобках пишется то, что будет предприниматься в случае выполнения условия. Далее else и снова в фигурных скобках то, что выполнится в случае, если условие не сработает. Количество различных действий между фигурными скобками неограниченно, фактически можно выполнить две различные программы. При сравнении можно использовать логические выражения.

Например:

```
<script language="JavaScript">
var x = 5;
var y = 10;
if (x>y) {
alert('x - максимальное число')
}
else
{
alert('y - максимальное число')
}
</script>
```

• оператор выбора switch

Это фактически несколько условных операторов, объединенных в одном. В данном операторе вычисляется одно выражение и сравнивается со значениями, заданными в блоках case. В случае совпадения выполняются операторы соответствующего блока case.

```
switch (выражение) {
case значение1:
оператор_1;
break;
case значение2:
оператор_2;
break;
.....
default:
оператор;
}
```

Если значение выражения не равняется ни одному из значений, заданных в блоках case, то вычисляется группа операторов блока default, если этот блок задан, иначе происходит выход из оператора switch.

Необязательный оператор break, задавемый в блоках case, выполняет безусловный выход их оператора switch.

2. Операторы цикла

Оператор цикла повторно выполняет последовательность операторов JavaScript, определенных в его теле, пока не выполниться некоторое заданное условие.

• цикл for (цикл со счетчиком)

```
for (i=1; i<10; i++){
<тело цикла>
}
```

Первый параметр (i=1) определяет счетчик и указывает его начальное значение. Этот параметр называется начальным выражением, поскольку в нем задается начальное значение счетчика (начальное значение в данном случае равно единице). Это выражение инициализации выполняется самым первым и всего один раз.

Второй параметр (i<10) - это условие, которое должно быть истинным, чтобы цикл выполнялся, как только условие цикла становится ложным, работа цикла завершается. Он называется условием цикла.

Проверка условия цикла осуществляется на каждом шаге; если условие истинно, то выполняется тело цикла (операторы в теле цикла). Цикл в данном случае выполнится только девять раз так как задано условие i<10.

Третий параметр (i++) - это оператор, который выполняется при каждом последовательном прохождении цикла. Он называется выражением инкремента, поскольку в нем задается приращение счетчика (приращение счетчика в данном случае равно единице). Пример автоматической прорисовки нескольких линий с помощью цикла for.

```
<script language="JavaScript" type="text/JavaScript">
for (var i=1; i<10; i++){
document.write("<hr align='center' width='100'>");
}
</script>
```

• цикл while (цикл с предусловием)

```
while (условие)
{
<тело цикла>
}
```

Пока значение условия - true (истинно), выполняется тело цикла. Тело цикла может быть представлено простым или составным оператором.

Оператор while содержит в скобках все необходимые параметры условия цикла (логическое выражение). После определения всех параметров цикла вводится открывающая фигурная скобка, символизирующая начало тела цикла. Закрывающая фигурная скобка вводится в конце тела цикла. Все операторы, введенные в скобках, выполняются при каждом прохождении цикла.

```
<script language="JavaScript">
var i=1;
while(i<=10){
document.write('число='+i+'<br>');
i=i+2;
}
</script>
```

• прерывание и перезапуск цикла

Оператор прерывания break позволяет прервать выполнение цикла и перейти к следующему за ним выражению:

```
a = 10;
i = 1;
while (a<100){
a = a * i;
if (i>4) break;
++i;
}
```

Если значение i превысит 4, то прерывается выполнение цикла.

Оператор перезапуска continue позволяет перезапустить цикл, т.е. оставить невыполненными все последующие выражения, входящие в тело цикла, и запустить выполнение цикла с самого начала.

```
a = 10;
i = 1;
while (a<100){
++i;
if (i>2 && i<11) continue;
a = a * i;
}
```

Создание и вызов функций в JavaScript

В JavaScript функцией называется именованная часть программного кода, которая выполняется только при обращении к ней посредством указания ее имени. Функции создаются с помощью ключевого слова function. Обычно функции располагают в секции <head>. Такое расположение функций в HTML-документе гарантирует их полную загрузку до того момента, когда их можно будет вызвать из секции <body>.

После названия функции (func_name) ставятся двойные круглые скобки, программный код при этом заключается в фигурные скобки:

```
<script language="JavaScript">
function func_name()
{
программный код функции (тело функции)
}
</script>
```

Для того, чтобы вызвать функцию в нужном месте, необходимо просто указать ее имя в тексте:

```
<script language="JavaScript">
func_name();
</script>
```

Второй вариант вызова функции непосредственно в HTML теге:

```
<a href="javascript:func_name()">Текст ссылки</a>
```

Ниже приведен код страницы HTML, после загрузки которой каждые три секунды будет появляться сообщение, генерируемое вызовом функции myMessage():

```
<script>
function myMessage()
{
alert("My Message")
}
</script>
<body onload='setTimeout ("myMessage()",3000)'>
<p>Каждые три секунды будет появляться сообщение</p>
</body>
```

Метод setTimeout() запускает выполнение кода JavaScript, задаваемого первым строковым параметром, через определенный промежуток времени после выполнения метода. Интервал задается в миллисекундах (1000 соответствует 1 секунде).

Примеры для реализации:

Существование программных объектов самих по себе не имеет никакого смысла. Они дадут преимущества при программировании тогда, когда можно организовать их взаимодействие.

```
<!DOCTYPE HTML>
<html>
<head>
<!-- Тег meta для указания кодировки -->
<meta charset="utf-8">
</head>
<body>
<p>Начало документа...</p>
<p>...Конец документа</p>
</body>
</html>
```

Этот пример использует следующие элементы:

<script> ... </script>

Тег script содержит исполняемый код. Предыдущие стандарты HTML требовали обязательного указания атрибута type, но сейчас он уже не нужен. Достаточно просто <script>. Браузер, когда видит <script>:

1. Начинает отображать страницу, показывает часть документа до script

2. Встретив тег script, переключается в JavaScript-режим и не показывает, а исполняет его содержимое.

3. Закончив выполнение, возвращается обратно в HTML-режим и *только тогда* отображает оставшуюся часть документа.

Попробуйте этот пример в действии, и вы сами всё увидите.

alert(сообщение)
Отображает окно с сообщением и ждёт, пока посетитель не нажмёт «Ок».

Если JavaScript-кода много — его выносят в отдельный файл, который подключается в HTML:

Здесь /path/to/script.js — это абсолютный путь к файлу, содержащему скрипт (из корня сайта). Браузер сам скачает скрипт и выполнит.

Можно указать и полный URL, например:

Вы также можете использовать путь относительно текущей страницы, например src="lodash.js" обозначает файл из текущей директории.

Чтобы подключить несколько скриптов, используйте несколько тегов:

Внешние скрипты, порядок исполнения

```
<script src="/path/to/script.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/lodash.js/3.2.0/lodash.js"></script>
...
```

Если указан атрибут src, то содержимое тега игнорируется.

В одном теге SCRIPT нельзя одновременно подключить внешний скрипт и указать код.
Вот так не сработает:

Нужно выбрать: либо SCRIPT идёт с src, либо содержит код. Тег выше следует разбить на два: один — с src, другой — с кодом, вот так:
Обычно тег <script> блокирует отображение страницы.

Например, в примере ниже — пока все кролики не будут посчитаны — нижний <p> не будет показан:

```
<script src="file.js">
alert(1); // так как указан src, то внутренняя часть тега игнорируется
</script>
<script src="file.js"></script>
<script>
alert( 1 );
</script>
```

Асинхронные скрипты: defer/async

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<p>Начинаем считать:</p>
<p>Кролики посчитаны!</p>
</body>
</html>
<script>
alert( 'Первый кролик!' );
alert( 'Второй кролик!' );
alert( 'Третий кролик!' );
</script>
```

Такое поведение называют «синхронным». Как правило, оно вполне нормально, но есть один нюанс.

Пока браузер не загрузит и не выполнит внешний скрипт, он не покажет часть страницы под ним.

В ряде случаев это совсем неуместно. Например, мы подключаем внешний скрипт, который показывает рекламу или вставляет счётчик посещений, а затем идёт наша страница. Конечно, неправильно, что пока счётчик или реклама не подгрузятся — оставшаяся часть страницы не показывается. Счётчик посещений не должен никак задерживать отображение страницы сайта. Реклама — это дополнение к странице, она не должна как-то тормозить сайт и нарушать его функционал.

А что, если сервер, с которого загружается внешний скрипт, перегружен? Посетитель в этом случае может ждать очень долго. Вот пример, с подобным скриптом (стоит искусственная задержка загрузки):

```
<p>Начало страницы...</p>
<p>...Важная информация!</p>
<script src="https://js.cx/hello/ads.js?speed=0"></script>
```

В примере выше важная информация не покажется, пока не загрузится внешний скрипт. Но действительно ли он так важен, что мы хотим заставить посетителя ждать? Если это реклама или счётчик посещаемости, то вряд ли.

Можно поставить все подобные скрипты в конец страницы — это уменьшит проблему, но не избавит от неё полностью, поскольку скриптов может быть несколько, и если какой-то один скрипт тормозит или завис, то последующие будут его ждать.

Кроме того, когда скрипты в конце страницы — то они начнут грузиться только тогда, когда вся страница загрузится. А это не всегда правильно, например счётчик посещений наиболее точно сработает, если загрузить его пораньше.

Кардинально решить эту проблему помогут атрибуты async или defer:

Атрибут async

Поддерживается всеми браузерами, кроме IE9-. Скрипт выполняется полностью асинхронно. То есть, при обнаружении <script async src="..."> браузер не останавливает обработку страницы, а спокойно работает дальше. Когда скрипт будет загружен — он выполнится.

Атрибут defer

Поддерживается всеми браузерами, включая самые старые IE. Скрипт выполняется асинхронно, не заставляет ждать страницу, но, в отличие от async, браузер гарантирует, что относительный порядок скриптов с defer будет сохранён. То есть, в таком коде (с async) первым сработает тот скрипт, который раньше загрузится: А в таком коде (с defer) первым сработает всегда 1.js, а скрипт 2.js, даже если загрузился раньше, будет его ждать.

```
<script src="1.js" async></script>
<script src="2.js" async></script>
<script src="1.js" defer></script>
<script src="2.js" defer></script>
```

Атрибут defer используют в тех случаях, когда второй скрипт 2.js зависит от первого 1.js, к примеру — использует что-то, описанное первым скриптом.

Тот же пример с async:

```
<p>Начало страницы...</p>
<p>...Важная информация!</p>
<script async src="https://js.cx/hello/ads.js?speed=0"></script>
```

При запуске вы увидите, что вся страница отобразилась тут же, а alert из внешнего скрипта появится позже, когда загрузится скрипт.

Общие задания к лабораторной работе:

1. Напишите скрипт, запрашивающий количество учащихся студенческой группы и в соответствии с этим количеством – их фамилии и имена. Фамилии и имена необходимо отобразить в окне браузера.

2. Напишите скрипт, который запрашивает фамилии и имена до тех пор, пока пользователь в окне ввода не нажмет кнопку "Отмена". Фамилии и имена необходимо отобразить в окне браузера.

3. Создайте скрипт, который выводил бы на странице горизонтальные линии разной длины;

4. Создайте скрипт, который выводил бы на странице таблицу умножения;

5. Создайте скрипт, который выводил бы на странице таблицу перевода температуры из градусов по шкале цельсия (c) в градусы по шкале фаренгейта (f) по формуле f = 1.8c + 32 для значений температуры от 0 до 30 oc с шагом 1 oc;

6. Создайте скрипт, который выводил бы на странице таблицу значений y для значений x, равных 0, 0.1, 0.2, ..., 2, если y задается формулой y = 4x2 – 5.5x + 2

7. Создайте скрипт, который выводил бы на странице таблицу значений факториала натурального числа n = 0, 1, 2, ..., 30 (факториал 0 = 1).

Индивидуальные задания по вариантам.

Вариант определяется следующим образом: 1-1, 2-2, 3-3, 4-4, 5-1, 6-2 и т.д.

Вариант № 1

1. Написать скрипт, в котором будет предлагаться ввести текст размером не менее случайного сгенерированного программно числа (генерировать в диапазоне от 0 до 50) символов. Поменять местами в строке первое и последнее слово и вывести в сообщении полученный результат.

2.Написать цикл, который предлагает через prompt ввести число. Умножить в цикле число на случайное число. Если полученное число меньше 100, попросить ввести еще раз и т.д.

3.Создать скрипт, который выводит все простые числа из интервала от 2 до 100. В результате полученные числа выводить одной строкой через запятую.

4.Написать скрипт, который будет запрашивать у пользователя ввод признаков описания грибов по цвету (белый, желтый, красный, коричневый) и наличию юбки (есть, нет). Далее использовать оператор switch, который будет для белого гриба без юбки выводить в сообщении «вы нашли белый гриб», для красного гриба с юбкой выводить «а не мухомор ли это», для красного гриба без юбки выводить «может, это подосиновик», для остальных выводить «с грибами надо быть внимательными».

5. Написать скрипт, который при наведении курсора на надпись изменяет цветовую гамму окна в желтую, при уводе курсора с надписи — возвращает прежнюю цветовую гамму окна.

6. Напишите код, который:

– запрашивает по очереди значения при помощи prompt и сохраняет их в массиве;
– заканчивает ввод, как только посетитель введет пустую строку, при этом ноль не должен заканчивать ввод;
– выводит сумму всех значений массива.

7. Вывести таблицу с цветом фона, подобным шахматной доске. Код оформить в виде функции, которая будет принимать три аргумента: fcolor — первый цвет таблицы, scolor — второй цвет ячеек таблицы, contents — массив, содержимое которого может выводиться в таблице.

8. Дан двумерный массив размером N на M. Задать значение элементов матрицы от –9 до 9 с помощью функции random(). Размеры матрицы ввести с помощью клавиатуры. Вывести матрицу (в виде таблицы). Найти разность между максимальным и минимальным элементами матрицы.

9. Дана строка на русском языке. Подсчитать количество гласных букв, содержащихся в строке. Регистр не учитывать.

10. Создать функцию func (arr, a, b), которая принимает массив чисел arr и возвращает новый массив, который содержит только числа из arr в диапазоне от a до b. Функция не должна менять arr.

Вариант № 2

1. Написать скрипт, в котором будет предлагаться ввести текст размером не менее случайного сгенерированного программно числа (генерировать в диапазоне от 10 до 100) символов. Затем ввести текст размером в три раза меньше предыдущего. Если второй введенный текст встречается в первом введенном тексте, то вывести позицию вхождения второй строки в первую. Иначе сообщить, что совпадения нет.

2. Написать цикл, который предлагает через prompt ввести один из вариантов предлагаемого текста (например, укажите пол: мужской/женский). Если посетитель ввел другое значение, попросить ввести еще раз и т.д. Цикл должен спрашивать, пока посетитель не введет нужное значение либо не нажмет кнопку Cancel (ESC).

3. Создать скрипт, который выводит все числа из интервала от 2 до 100, которые делятся на 3 и 17 с остатком 1. В результате полученные числа выводить одной строкой через запятую.

4. Написать скрипт, который будет запрашивать у пользователя ввод сведений о погоде (дождливо, ветрено, солнечно, снежно и т.п.). Далее использовать оператор switch, который будет для «дождливо» выводить в сообщении «захватите зонт и наденьте калоши», для «ветрено» выводить «укутайтесь по- теплее», для «солнечно» выводить «возьмите солнечные очки — берегите глаза», для «снежно» — «пора лепить снеговиков», во всех остальных случаях вы- водить «одевайтесь, как хотите, но погода непредсказуема».

5. Создать скрипт, который при наведении курсора на надпись выводит в цикле сообщение с просьбой ввести ключевое слово, когда ключевое слово (выбрано самостоятельно) будет введено, то выйти из цикла, при уводе курсора с надписи — выводить сообщение с помощью «Курсор вне зоны надписи».

6. Напишите код, который:

– запрашивает по очереди значения при помощи prompt и сохраняет их в массиве;

– заканчивает ввод, как только посетитель нажмет кнопку Отмена, при этом ноль не должен заканчивать ввод;

– выводит разницу между четными и нечетными значениями массива.

7. Дан двумерный массив размером N на M. Задать значение элементов матрицы от –9 до 9 с помощью функции random(). Размеры матрицы ввести с помощью клавиатуры. Вывести матрицу (в виде таблицы). Найти сумму отрицательных элементов матрицы.

8. Дана строка. Подсчитать количество цифр, содержащихся в ней.

9. Написать функцию Graf(), которая будет принимать неограниченное количество аргументов (целых чисел) и строить разноцветный график.

10. На входе массив чисел, например, arr = [1,2,3,4,5]. Написать функцию getSums(arr), которая

возвращает массив его частичных сумм, т.е.:

arr=[1, 2, 3] -> getSums(ar)=[1, 1+2, 1+2+3] = [1, 3, 6]

Вариант №3.

1. Написать скрипт, в котором будет предлагаться ввести текст размером не менее случайного сгенерированного программно числа (генерировать в диапазоне от 1 до 45) символов. Все четные символы перевести в верхний регистр, все нечетные — в нижний регистр. Вывести в сообщении полученный результат.

2. Написать цикл, который предлагает через prompt ввести число (в диапазоне от 15 до 90) и некоторый текст. Если посетитель ввел другое число или текст длиной меньше 10, то попросить ввести еще раз, и т.д. Цикл должен спрашивать число и текст, пока либо посетитель не введет требуемые данные, либо не нажмет кнопку Cancel (ESC).

3. Создать скрипт, который выводит все числа из интервала от 2 до 100, которые делятся на 5 и 11 с остатком 2. В результате полученные числа выводить одной строкой через запятую.

4. Написать скрипт, который будет запрашивать у пользователя ввод времени в формате HH:MM (часы:минуты). Далее использовать оператор switch, который будет для времени с 07:00 до 12:00 выводить в сообщении «утро», с 12:00 до 17:00 выводить «день», с 17:00 до 21:00 выводить «вечер», с 21:00 до 07:00 — «ночь».

5. Создать скрипт, который при наведении курсора на надпись выводит в цикле сообщение с просьбой ввести результат умножения двух случайно определенных чисел в диапазоне от 1 до 10, если результат ввели правильно, то выйти из цикла, при уводе курсора с надписи — выводить сообщение «Курсор вне зоны надписи».

6. Напишите код, который:

– запрашивает по очереди значения при помощи prompt и сохраняет их в массиве;

– заканчивает ввод, как только посетитель введет пустую строку, при этом ноль не должен заканчивать ввод;

– выводит произведение всех значений массива.

7. Дан двумерный массив размером N на M. Задать значение элементов матрицы от -9 до 9 с помощью функции random(). Размеры матрицы ввести с помощью клавиатуры. Вывести матрицу (в виде таблицы). Найти максимум из сумм элементов строк.

8. Дана строка-предложение. Подсчитать количество слов.

9. Написать функцию First(str), которая возвращает строку str с заглавным первым символом.

Вариант № 4

1. Написать скрипт, в котором будет предлагаться ввести текст размером не менее случайного сгенерированного программно числа (генерировать в диапазоне от 10 до 100) символов. Определить количество символов во введенном тексте, наличие вхождений таких стоп-слов, как «был», «имеет», «есть». Вывести общим сообщением информацию о длине текста и вхождениях указанных слов.

2. Написать цикл, который предлагает через prompt ввести число больше 100 и некоторый текст. Если посетитель ввел другое число, то попросить ввести еще раз, и т.д. Цикл должен спрашивать число, пока посетитель не введет требуемые данные либо не нажмет кнопку Cancel (ESC). При выходе из цикла вывести второе слово из введенного текста или сообщить, что текст состоит из одного слова.

3. Создать скрипт, который выводит все числа из интервала от 2 до 100, которые делятся на 13 и 7 с остатком 3. В результате полученные числа выводить одной строкой через запятую.

4. Написать скрипт, который будет запрашивать у пользователя ввод даты. Далее использовать оператор switch, который будет для 1, 2 и 12 месяцев выводить в сообщении «зима», для 3, 4, 5 месяцев выводить «весна», для 6, 7, 8 месяцев выводить «лето», для 9, 10, 11 месяцев — «осень».

5. Создать скрипт, который при наведении курсора на надпись выводит в цикле сообщение с просьбой ввести остаток от деления двух случайно определенных чисел в диапазоне от 10 до 30, если результат ввели правильно, то выйти из цикла, при уводе курсора с надписи — выводить сообщение «Курсор вне зоны надписи».

6. Напишите код, который:

– запрашивает по очереди значения при помощи prompt и сохраняет их в массиве;
– заканчивает ввод, как только посетитель введет пустую строку, не число или нажмет кнопку Отмена, при этом ноль не должен заканчивать ввод;
– выводит все значения массива.

7. Дан двумерный массив размером N на M. Задать значение элементов матрицы от –9 до 9 с помощью функции random(). Размеры матрицы ввести с помощью клавиатуры. Вывести матрицу (в виде таблицы). Найти минимум из сумм элементов столбцов.

8. Дана строка на английском языке. Подсчитать количество гласных прописных букв, содержащихся в строке.

9. Создать функцию sea(arr, value), которая ищет в массиве arr значение value и возвращает его номер, если найдено, или «в массиве нет данного элемента», если не найдено.

10. Создать функцию func(str, maxlength), которая проверяет длину строки str, и если она превосходит maxlength — заменяет конец str на "...", так, чтобы ее длина стала равна maxlength. Результатом функции должна быть (при необходимости) усеченная строка.

Дополнительное задание для студентов, сдавших лабораторную работу не в срок.

1. Вводится произвольное целое число. Определить является ли число палиндромом. (Например: 10101 — да, 101001 — нет).

2. Написать функцию Sum(n), которая для данного n вычисляет сумму чисел от 1 до n. Сделать три варианта решения: с использованием цикла, через рекурсию и используя арифметическую прогрессию.

3. Создать метод у всех числовых объектов, который бы выводил значение числа от 0 до 99 в виде слов. Например, число 45 должно выводиться как «сорок пять».

4. Написать функцию camelize(str), которая преобразует строки вида «myshort-string» в «myShortString», т.е. дефисы удаляются, а все слова после них получают заглавную букву.

5. Написать функцию func(arr), которая возвращает массив, содержащий только уникальные элементы arr.

6. Определить количество дней в месяце 2021 г. С клавиатуры вводится номер месяца, на экран выдается сообщение, например, «в апреле 30 дней».