Частное учреждение образования

«Колледж бизнеса и права»

|  |
| --- |
| УТВЕРЖДАЮ  Заведующий  методическим кабинетом  \_\_\_\_\_\_\_\_\_\_Е.В.Паскал  «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ |

|  |  |
| --- | --- |
| Специальность:  2-40 01 01 «Программное обеспечение информационных технологий» | Дисциплина: «Скриптовые языки программирования» |

**Лабораторная работа № 7**

**Инструкционно-технологическая карта**

Тема: Циклы while, do…while, for. Прерывание цикла «break». Переход к следующей итерации continue.

Цель: Научится пользоваться while, do…while, for, использовать оператор прерывания цикла break, оператора переход к следующей итерации continue.

Время выполнения: 2 часа

# Порядок выполнения работы

1. Изучить теоретические сведения к лабораторной работе.
2. Реализовать алгоритм решения задачи.
3. Отлаженную, работающую программу сдать преподавателю. Работу программы показать с помощью самостоятельно разработанных тестов.
4. Ответить на контрольные вопросы.

# Теоретические сведения

При написании скриптов зачастую встаёт задача сделать однотипное действие много раз.

Например, вывести товары из списка один за другим. Или просто перебрать все числа от 1 до 10 и для каждого выполнить одинаковый код.

Для многократного повторения одного участка кода предусмотрены циклы.

## Цикл «while»

Цикл **while** имеет следующий синтаксис:

while (condition) {

    // код

    // также называемый "телом цикла"

}

Код из тела цикла выполняется, пока условие **condition** истинно.

Например, цикл ниже выводит i, пока i < 3:

let i = 0;

while (i < 3) { // выводит 0, затем 1, затем 2

    alert(i);

    i++;

}

Одно выполнение тела цикла по-научному называется итерация. Цикл в примере выше совершает три итерации.

Если бы строка i++ отсутствовала в примере выше, то цикл бы повторялся (в теории) вечно. На практике, конечно, браузер не позволит такому случиться, он предоставит пользователю возможность остановить «подвисший» скрипт, а JavaScript на стороне сервера придётся «убить» процесс.

Любое выражение или переменная может быть условием цикла, а не только сравнение: условие while вычисляется и преобразуется в логическое значение.

Например, while (i) – более краткий вариант while (i != 0):

let i = 3;

while (i) { // когда i будет равно 0, условие станет ложным, и цикл остановится

    alert(i);

    i--;

}

## Цикл «do…while»

Проверку условия можно разместить под телом цикла, используя специальный синтаксис **do..while**:

do {

    // тело цикла

} while (condition);

Цикл сначала выполнит тело, а затем проверит условие condition, и пока его значение равно **true**, он будет выполняться снова и снова.

Например:

let i = 0;

do {

    alert(i);

    i++;

} while (i < 3);

Такая форма синтаксиса оправдана, если нужно, чтобы тело цикла выполнилось хотя бы один раз, даже если условие окажется ложным. На практике чаще используется форма с предусловием: while (…) {…}.

## Цикл «for»

Более сложный, но при этом самый распространённый цикл – цикл for.

Выглядит он так:

for (начало; условие; шаг) {

    // ... тело цикла ...

}

Давайте разберёмся, что означает каждая часть, на примере. Цикл ниже выполняет **alert(i)** для **i** от **0** до (но не включая) **3**:

for (let i = 0; i < 3; i++) { // выведет 0, затем 1, затем 2

    alert(i);

}

Рассмотрим конструкцию **for** подробней:

| **Часть** | **Написание** | **Описание** |
| --- | --- | --- |
| начало | i = 0 | Выполняется один раз при входе в цикл |
| условие | i < 3 | Проверяется перед каждой итерацией цикла. Если оно вычислится в false, цикл остановится. |
| шаг | i++ | Выполняется после тела цикла на каждой итерации перед проверкой условия. |
| тело | alert(i) | Выполняется снова и снова, пока условие вычисляется в true |

В целом, алгоритм работы цикла выглядит следующим образом:

Выполнить \* начало \*

→ (Если \* условие \* == true → Выполнить \* тело \*, Выполнить \* шаг \*)

→ (Если \* условие \* == true → Выполнить \* тело \*, Выполнить \* шаг \*)

→ (Если \* условие \* == true → Выполнить \* тело \*, Выполнить \* шаг \*)

→ ...

То есть, начало выполняется один раз, а затем каждая итерация заключается в проверке условия, после которой выполняется тело и шаг.

Вот в точности то, что происходит в нашем случае:

// for (let i = 0; i < 3; i++) alert(i)

// Выполнить начало

let i = 0;

// Если условие == true → Выполнить тело, Выполнить шаг

if (i < 3) { alert(i); i++ }

// Если условие == true → Выполнить тело, Выполнить шаг

if (i < 3) { alert(i); i++ }

// Если условие == true → Выполнить тело, Выполнить шаг

if (i < 3) { alert(i); i++ }

// ...конец, потому что теперь i == 3

## Прерывание цикла: «break»

Обычно цикл завершается при вычислении условия в **false**.

Но также можно выйти из цикла в любой момент с помощью специальной директивы **break**.

Например, следующий код подсчитывает сумму вводимых чисел до тех пор, пока посетитель их вводит, а затем – выдаёт:

let sum = 0;

while (true) {

    let value = +prompt("Введите число", '');

    if (!value) break; // (\*)

    sum += value;

}

alert('Сумма: ' + sum);

Директива **break** в строке (\*) полностью прекращает выполнение цикла и передаёт управление на строку за его телом, то есть на **alert**.

Вообще, сочетание «бесконечный цикл + **break**» – отличная штука для тех ситуаций, когда условие, по которому нужно прерваться, находится не в начале или конце цикла, а посередине.

## Переход к следующей итерации: continue

Директива **continue** – «облегчённая версия» **break**. При её выполнении цикл не прерывается, а переходит к следующей итерации (если условие все ещё равно true).

Её используют, если понятно, что на текущем повторе цикла делать больше нечего.

Например, цикл ниже использует **continue**, чтобы выводить только нечётные значения:

for (let i = 0; i < 10; i++) {

    // если true, пропустить оставшуюся часть тела цикла

    if (i % 2 == 0) continue;

    alert(i); // 1, затем 3, 5, 7, 9

}

Для чётных значений **i**, директива continue прекращает выполнение тела цикла и передаёт управление на следующую итерацию **for** (со следующим числом). Таким образом **alert** вызывается только для нечётных значений.

# Контрольные вопросы

1. Опишите синтаксис всех видов цикла.
2. Какие части могут отсутствовать в цикле for?
3. Для чего предназначена директива break?
4. Для чего предназначена директива continue?

# Варианты индивидуальных заданий

**Задание 1**

1. Проверить является ли данная числовая последовательность а1, a2,..., an возрастающей.
2. Информация о температуре воздуха за месяц задана в виде массива. Определить, сколько раз температура опускалась ниже 0оC. Число дней конкретного месяца ввести с клавиатуры.
3. Информация о среднесуточной температуре воздуха за месяц задана в виде массива. Определить, сколько дней температура была ниже среднесуточной.
4. Дан числовой массив А, состоящий из n-элементов. Найти среднее арифметической положительных элементов этого массива.
5. Дан числовой массив А, состоящий из n-элементов. Все положительные элементы этого массива уменьшить на 0.5.
6. Дан числовой массив А, состоящий из n-натуральных чисел. Определить количество элементов массива больше заданного числа.
7. Дан числовой массив А, состоящий из n-натуральных чисел. Определить количество элементов массива, являющихся нечетными числами.
8. Дан числовой массив А, состоящий из n-натуральных чисел. Определить количество элементов массива, являющихся кратными 7.
9. Дан числовой массив А, состоящий из n-натуральных чисел. Определить количество элементов массива кратных 3, но не кратных 5.
10. Дан числовой массив А, состоящий из n-натуральных чисел. Определить количество элементов массива, удовлетворяющих условию А[i] < (A[i-1] + A[i+1])/2.
11. Дан числовой массив А, состоящий из n-натуральных чисел. Определить количество элементов массива, удовлетворяющих условию 2\*k<A[k]<3\*k.
12. Дан числовой массив А, состоящий из n-натуральных чисел. Определить количество элементов массива, имеющих четные порядковые номера и являющихся нечетными числами.
13. Дан числовой массив А, состоящий из n-натуральных чисел. Определить количество элементов массива, которые при делении на 7 дают остаток 1, 2 или 5.
14. Имеется n итоговых оценок студента. Определить является ли студент отличником.
15. Имеется n итоговых оценок студента. Определить является ли студент неуспевающим.
16. Имеется n итоговых оценок студента. Подсчитать количество двоек, троек, четверок и пятерок.
17. Дана последовательность х1, х2, …, хn, упорядоченная в порядке возрастания и вещественное у. Найти такое k, что x[k] < y x[k].
18. Имеется n итоговых оценок студента. Расположить эти оценки в следующем порядке: пятерки, двойки, четверки, тройки.
19. Дана последовательность х1, х2, …, хn. Подсчитать количество таких троек, что xi-1<xi<xi+1 (i=2,n-1).
20. Дана последовательность х1,х2,…,хn. Найти номер элемента, который отличается от среднего арифметического значения элементов последовательности на минимальную величину.
21. Дана последовательность х1,х2,…,хn. Найти наибольшую сумму подряд идущих элементов.
22. Дана последовательность х1,х2,…,хn. Подсчитать количество элементов последовательности, больших среднего арифметического значения положительных элементов последовательности.

**Задание 2.**

1. Дана последовательность натуральных чисел х1,х2,…,хn. Изменить данную последовательность так, чтобы в начале стояли все четные, а затем нечетные элементы последовательности.
2. Дана последовательность натуральных чисел х1,х2,…,хn. Изменить данную последовательность так, чтобы в начале стояли все нулевые элементы, затем отрицательные, а затем положительные элементы последовательности.
3. Даны две последовательности чисел а1,а2,…,аn и b1,b2,…,bn. Подсчитать, какое число раз встречается ситуация, когда а[i]>b[i] и а[i+1]<b[i+1] (i=1,n-1).
4. Даны две последовательности чисел а1,а2,…,аn и b1,b2,…,bn. Найти i, при котором а[i]+b[i] наибольшая из всех таких пар.
5. Даны две последовательности чисел а1,а2,…,аn и b1,b2,…,bn. Найти количество пар а[i],b[i] (i=1,n) таких, что оба числа в паре четные.
6. Дана последовательность х1,х2,…,хn. Найти наибольший по модулю элемент последовательности с указанием его номера.
7. Дана последовательность х1,х2,…,хn. Поменять местами самый большой элемент с самым маленьким.
8. Дана последовательность х1,х2,…,хn. Найти самый большой по модулю отрицательный элемент.
9. Дана последовательность х1,х2,…,хn и число у. Подсчитать количество элементов последовательности, которые меньше у.
10. Проверить является ли данная числовая последовательность а1, a2,..., an убывающей;
11. Проверить является ли данная числовая последовательность а1, a2,..., an невозрастающей.
12. Проверить является ли данная числовая последовательность а1, a2,..., an неубывающей.
13. Проверить является ли данная числовая последовательность а1, a2,..., an постоянной (все члены равны между собой).
14. Дана последовательность целых чисел a1, a2,...,an. Найти модуль наибольшего по модулю элемента последовательности.
15. Дана последовательность целых чисел a1, a2,...,an. Найти модуль наибольшего элемента последовательности.
16. Дана последовательность целых чисел a1, a2,...,an. Найти модуль наименьшего по модулю элемента.
17. Дана последовательность целых чисел a1, a2,...,an. Найти наименьший среди положительных элементов.
18. Дана последовательность целых чисел a1, a2,...,an. Найти наибольший среди отрицательных элементов.
19. Дана последовательность целых чисел a1, a2,...,an. Найти наибольший среди элементов, стоящих на четных местах.
20. Дана последовательность действительных чисел a1, a2,...,an. Найти сумму первых чисел, произведение которых меньше заданного числа b.
21. Дана последовательность действительных чисел a1, a2,...,an. Получить из заданной последовательности другую, в которой сначала бы шли отрицательные члены исходной последовательности, а затем неотрицательные.
22. Дана последовательность действительных чисел a1, a2,...,an. Найти сумму членов, удовлетворяющих условию |a[k]|>k.
23. Даны координаты x[i] и y[i] n-точек на плоскости. Подсчитать количество точек, лежащих в первой четверти.

# Литература

**Диков, А.В.** Клиентские технологии веб программирования: JavaScript и DOM: учебное пособие / А.В. Диков. – СПб: Лань, 2020 – 124 с.

**Читанамбри, Кирупа.** Изучаем React / Кирупа Читанамбри – 2-е изд. – М.: Эксмо, 2019. 368 с.

**Хавербеке, Марейн.** Выразительный JavaScript. Современное веб-программирование. / Марейн Хавербеке – 3-е изд. – СПб.: Питер, 2019. – 480 с.:

Преподаватель Рогалевич А.В.

Рассмотрено на заседании цикловой

комиссии программного обеспечения

информационных технологий №10

Протокол № \_\_\_ от \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Председатель ЦК\_\_\_\_\_\_\_\_ В.Ю.Михалевич