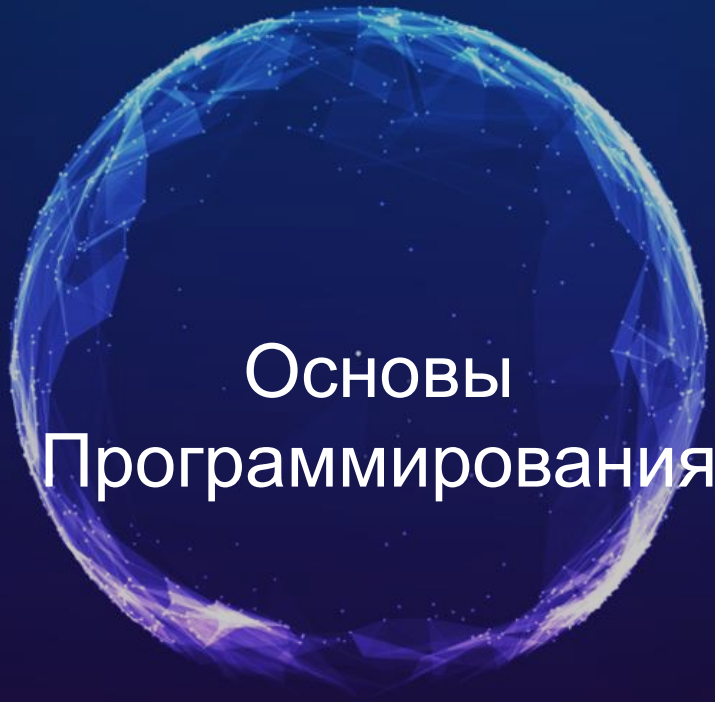


Основы Программирования

Занятие №4
Циклы

IT Education Academy

WWW.ITEA.UA

План занятия

- Цикл с предусловием
- Цикл с постусловием
- Цикл со счетчиком
- Операторы преждевременного выхода из цикла

Повторение материала

Циклы

Очень часто в программировании необходимо повторять одни и те же действия много раз.
Для описания этих повторений используются конструкции под названием циклы.

Цикл состоит из:

- тела - последовательности команд, которые необходимо выполнить
- условия окончания - выражение, определяющее, будет ли выполнено тело в очередной раз

Одно выполнение тела цикла называется итерацией.

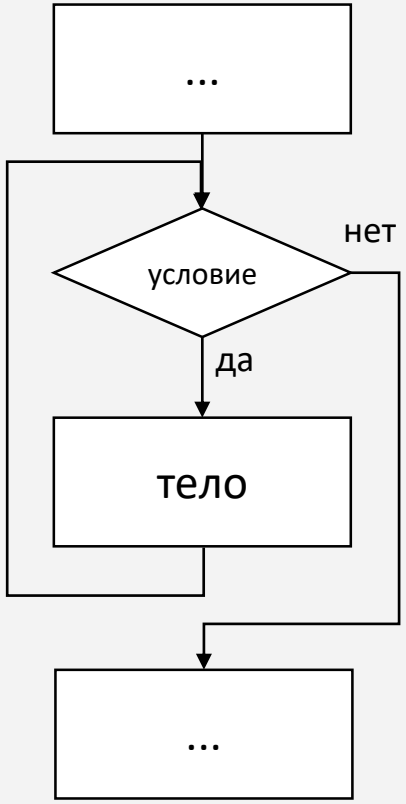
Цикл с предусловием

Условие проверяется перед выполнением тела.
Означает что тело может ни разу не выполниться.

В JavaScript для описания данного цикла используется конструкция:

```
while (условие)
  команда
```

```
Например:
var a = 1;
while (a < 10) {
  console.log(a);
  a++
}
console.log('end');
```



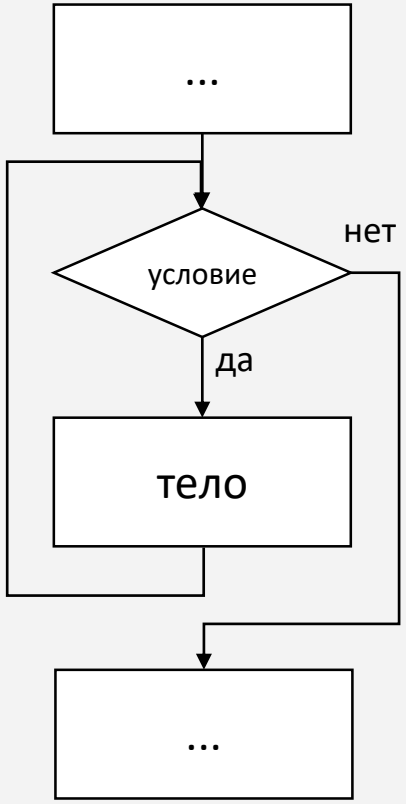
Цикл с предусловием

Условие проверяется перед выполнением тела.
Означает что тело может ни разу не выполниться.

В JavaScript для описания данного цикла используется конструкция:

```
while (условие)
  команда
```

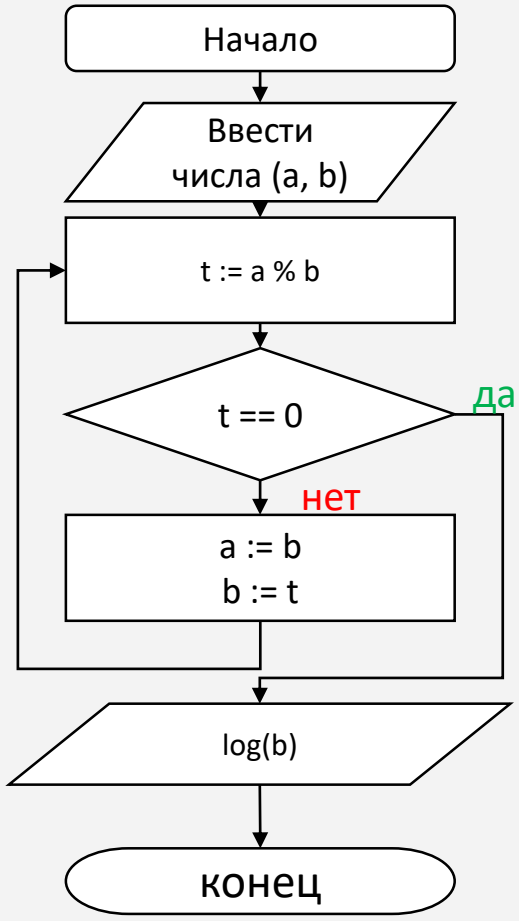
```
Например:
var a = 1;
while (a < 10) {
  console.log(a);
  a++
}
console.log('end');
```



Цикл с предусловием

Задача.

1. Вывести на экран числа от 1 до 20 (включая).
2. Вывести эти же числа, но дополнительно писать напротив каждого числа четное/нечетное.
   Например:
   1 - нечетное
   2 - четное
   3 - нечетное
   ...
3. Реализовать алгоритм поиска наибольшего общего делителя.



Цикл в цикле

В теле цикла может быть записана любая языковая конструкция. Цикл не является исключением. Поэтому в теле одного цикла можно использовать еще один. Например,

```
var i = 0;
while (i < 3) {
  var j = 0;
  while (j <= 3) {
    // сколько раз выполниться этот блок?
    j++
  }
  i++
}
```

Цикл в цикле

Задача.
Вывести на экран табличку умножения 10х10

Цикл с постусловием

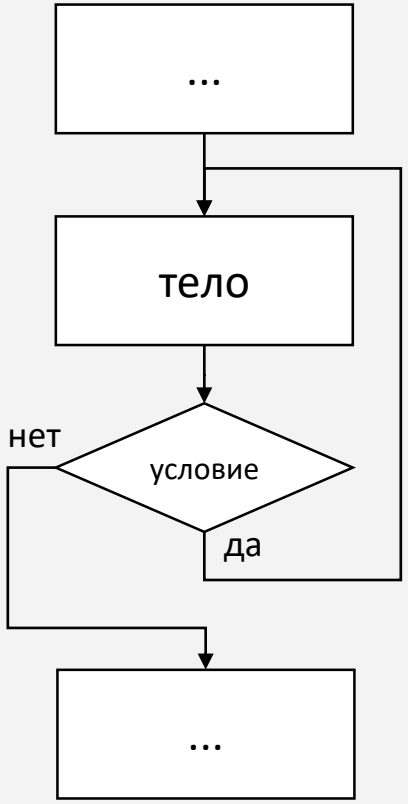
Условие проверяется после выполнением тела.
Означает что тело выполнится хотя бы один раз.

В JavaScript для описания данного цикла используется конструкция:

```
do
  команда;
while (условие);
```

Например:

```
var a = 1;
do {
  console.log(a);
  a++;
} while (a < 10);
console.log('end');
```



Цикл с постусловием

Задача.

Вывести на экран числа от 1 до 20 (включая), используя цикл с постусловием.

Действия пользователя

В условиях цикла может также использоваться ввод пользователя. Например:

```
var name = prompt('Как Вас зовут? (для выхода - никак)');
while (name != 'никак') {
  alert('Привет, ' + name + '!');
  name = prompt('Как Вас зовут? (для выхода - никак)');
}
```

Цикл со счетчиком

Синтаксис:

```
for (инициализация; условие; шаг)
  команда
```

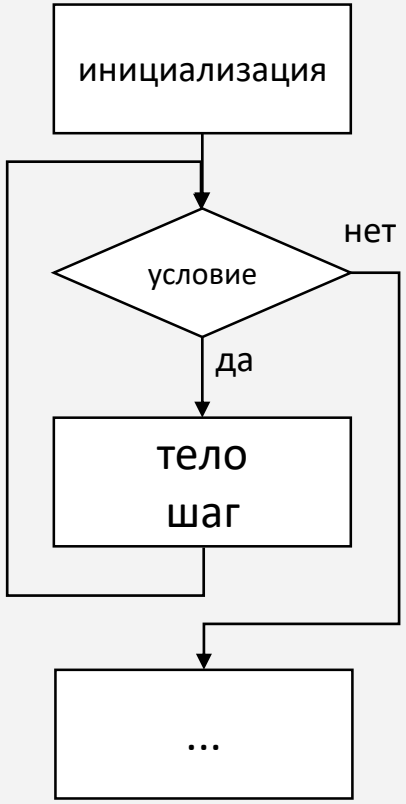
Например:

```
for (var i = 0; i < 10; i++) {
  console.log(i);
}
```

Любой цикл for можно переписать в while:

```
var i = 0;
while (i < 10) {
  console.log(i);
  i++;
}
```

Эти два примера одинаковые.



Цикл for

Задача.

1. Вывести на экран табличку умножения 10x10, используя циклы for.
2. Вывести только четные числа от 2 до 10.

Оператор break

Оператор break может использоваться только внутри циклов (на любом уровне вложенности).
Он позволяет заканчивать выполение самого глубокого цикла преждевременно.

```
for (var i = 0; i < 100; i++) { //цикл 1
  for (var j = 0; j < 100; j++) { //цикл 2
    if (i == j)
      break;  // по условию заканчивать цикл 2. НО продолжать выполнять цикл 1
    else
      console.log(i + ‘ != ‘ + j);
  }
}
```

Рекомендуется избегать использования этого оператора, а стараться указывать более точное условие выхода. Например в данном случае можно было бы написать условие:
`for (…; (j < 100) && (i != j); …)`

Оператор continue

Как и оператор break, continue может использоваться только внутри циклов (на любом уровне вложенности).
Он позволяет заканчивать текущее выполение итерации и переходить к следующей.

```
for (var i = 0; i < 100; i++) { //цикл 1
  for (var j = 0; j < 100; j++) { //цикл 2
    if (i == j)
      continue;  // по условию закончить итерацию в цикле 2 и продолжить со следующего значения
    else
      console.log(i + ' != ' + j);
  }
}
```

Рекомендуется избегать использования этого оператора, например:

```
for (...) { // цикл 2
  if (i != j) console.log(…);
}
```

Этот код идентичен.

Summary

1. Циклы нужны для описания повторяющихся действий.
2. Цикл состоит из условия выполения и тела.
3. Одно выполнения тела - итерация.
4. Цикл с предусловием (while) проверяет условие перед выполнением тела.
5. Цикл с постусловием (do-while) проверяет условие после выполнения тела.
6. Цикл for - цикл со счетчиком. Используется когда известно количество повторений. Может быть переписан в while.
7. Существуют дополнительные конструкции управления циклами - операторы break и continue.

Домашнее задание

1. Реализовать описанный в Д31 алгоритм возведения в степень. Протестировать работу для пар чисел (2,8), (3, 3), (5, 3), (4, 5).
2. Написать программу, которая ожидает в цикле просит пользователя ввести число до тех пор, пока пользователь не введет слово «стоп». По окончанию, программа должна выводить среднее значение (сумма всех чисел / кол-во введенных чисел). Проверить работу при вводе 5 чисел, 3 чисел, и если пользователь сразу введет «стоп».
3. Усовершенствовать программу, чтобы она выводила также минимальное число.
4. Написать программу (в NodeJS), которая выводит на экран следующий рисунок -> т.е. элементы по диагонали - 1, все остальные 0. Программа должна работать для любого N, где N - это размер рисунка. Подсказка: диагональный элемент, это тот элемент, у которого i == j, либо (N - i) == j (это условие в зависимости от выбранной системы отсчета, но суть одинаковая).

```
1000001
0100010
0010100
0001000
0010100
0100010
1000001
```

Спасибо за внимание!