loop

Введение

Циклы используются для того, чтобы некоторый участок кода выполнился несколько раз подряд.

Циклы могут повторять некоторый код заданное количество раз. Каждый такой проход цикла называется **итерацией** цикла.

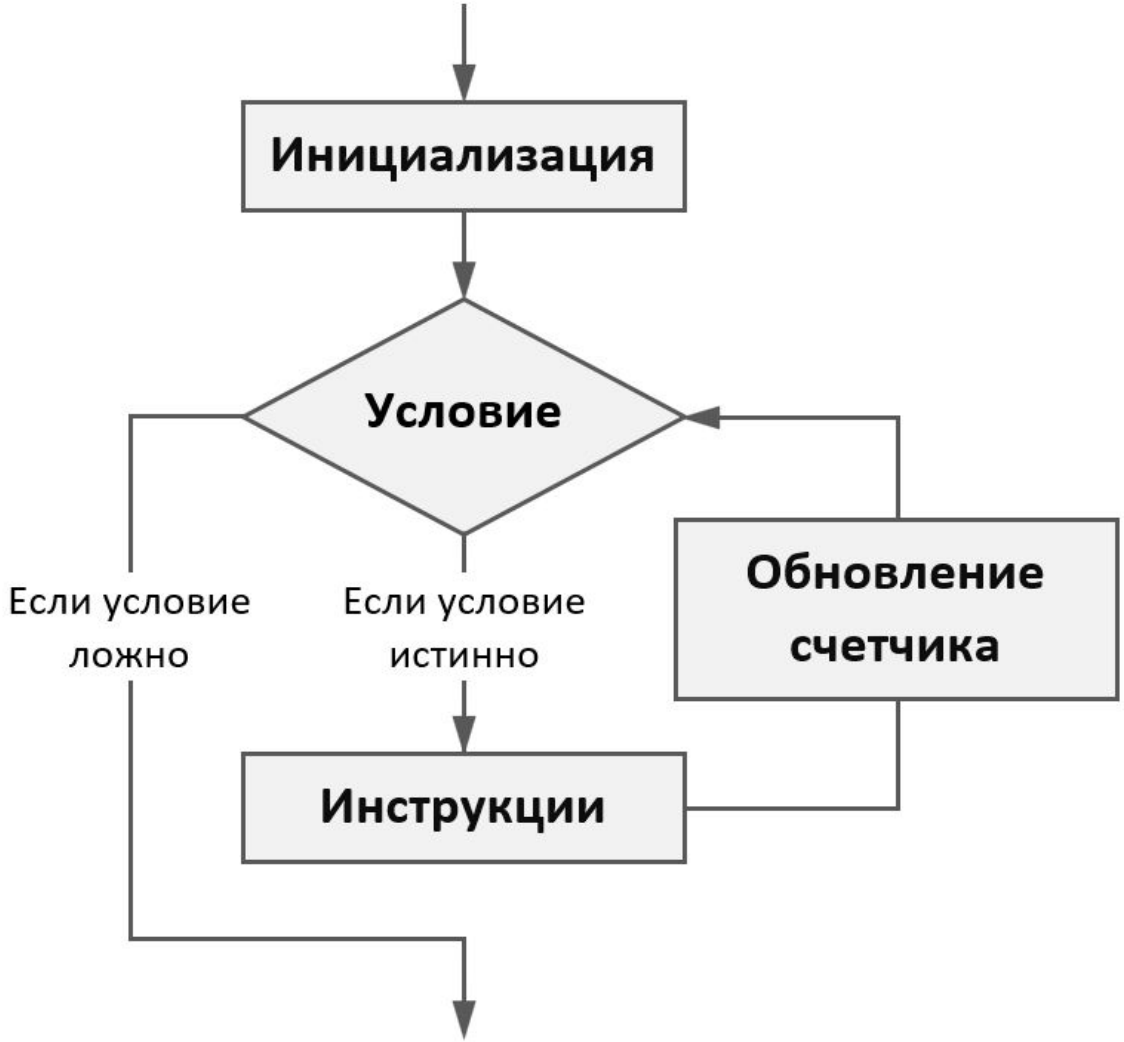
В циклах часто используются специальные переменные, которые каждую итерацию увеличивают свое значение на единицу. Такие переменные называются счетчиками циклов. Счетчики используются для того, чтобы подсчитывать, сколько раз выполнился цикл. Для счетчиков принято использовать буквы i, j и k.

Цикл for

Данный цикл в основном используется когда известно точное количество повторений. Этот цикл ещё называют циклом со счётчиком.

Синтаксис цикла «for»:

```
for (инициализация; условие; финальное выражение) {
  /* тело цикла */
 }
```


Инициализация
Условие
Если условие ложно
Если условие истинно
Обновление счетчика
Инструкции

Основные части конструкции цикла «for»:

- **инициализация** - это выражение, которое выполняется один раз перед выполнением цикла; обычно используется для инициализации счётчика;
- **условие** - это выражение, истинность которого проверяется перед каждой итерацией; если выражение **вычисляется как истина**, то выполняется итерация; в противном случае цикл «for» завершает работу;
- **финальное выражение** - это выражение, которое выполняется в конце каждой итерации; обычно используется для изменения счетчика;
- **тело цикла** - инструкции, выполнение которых нужно повторять.

Рассмотрим пример цикла, который выведет в консоль числа от 1 до 5:

```
// цикл «for» от 1 до 5, с шагом 1

for (let i = 1; i <= 5; i++) {
    console.log(i);
}
```

```
1
2
3
4
5
```

Как это работает: по шагам

Итерация	Переменные	Условие `i <= n`	Что происходит
1	i = 1 n = 5	true	Выводится `1`. Переменная i увеличивается на 1 — до 2.
2	i = 2 n = 5	true	Выводится `2`. Переменная i увеличивается на 1 — до 3.
3	i = 3 n = 5	true	Выводится `3`. Переменная i увеличивается на 1 — до 4.
4	i = 4 n = 5	true	Выводится `4`. Переменная i увеличивается на 1 — до 5.
5	i = 5 n = 5	true	Выводится `5`. Переменная i увеличивается на 1 — до 6.
6	i = 6 n = 5	false	Цикл останавливается.

Рассмотрим пример цикла, который выведет в консоль числа от 1 до 5:

```
// цикл «for» от 1 до 5, с шагом 1
let n = 5;

for (let i = 1; i <= n; i++) {
    console.log(i);
}
```

```
1
2
3
4
5
```

Задания

1. Выводим на экран сумму n-первых чисел
2. Вывести все чётные числа в диапазоне от 1 до 8
3. Вывести все числа от 1 до 100, которые кратные указанному пользователем числу, например 10
4. Вывести каждый 4-й элемент из указанного пользователем диапазона. Пользователь указывает минимальное и максимальное значения диапазона.

5. *Вывести все делители числа и подсчитать их количество

6. *Запросить число и проверить, простое ли оно или составное. Простое число делится без остатка только на себя и на единицу.

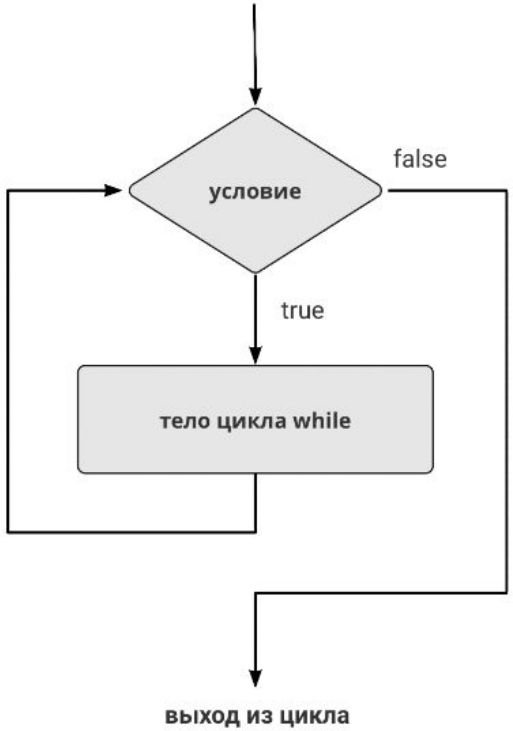
Цикл while (с предусловием)

Синтаксис

```
while (условие) {

    // тело цикла

}
```



Как это работает

1. Цикл while оценивает условие.
2. Если условие — true, выполняется код внутри цикла while.
3. Условие снова оценивается.
4. Этот процесс продолжается до тех пор, пока условие не станет false.
5. Когда условие становится false, цикл останавливается.

Пример 1. Выводим на экран числа от 1 до 5

```
// инициализируем переменные i и n
let i = 1, n = 5;
while (i <= n) {// цикл будет идти, пока i <= 5
    console.log(i);
    i += 1;// на каждой итерации i увеличивается на 1
 }
```

```
1
2
3
4
5
```

Как это работает

Итерация	Переменные	Условие i <= n	Что происходит
1	i = 1 n = 5	true	На экран выводится число 1. i увеличивается на 1 — до 2
2	i = 2 n = 5	true	На экран выводится число 2. i увеличивается на 1 — до 3
3	i = 3 n = 5	true	На экран выводится число 3. i увеличивается на 1 — до 4
4	i = 4 n = 5	true	На экран выводится число 4. i увеличивается на 1 — до 5
5	i = 5 n = 5	true	На экран выводится число 5. i увеличивается на 1 — до 6
6	i = 6 n = 5	false	Выход из цикла

Пример 2. Найти сумму положительных чисел. Пользователь вводит числа с клавиатуры и они суммируются. Если пользователь введет отрицательное число или нажмет отмена, то подсчет останавливается

1. Определить количество цифр в введенном числе
2. Посчитать факториал введенного пользователем числа.
3. Запросить 2 числа и найти все общие делители.

Какое последнее значение выведет код?

```
let i = 3;


while (i) {
  console.log( i-- );
}
```

```
let i = 3;


while (i) {
  console.log( --i );
}
```

Проверка `while(i)` остановит цикл при `i = 0`

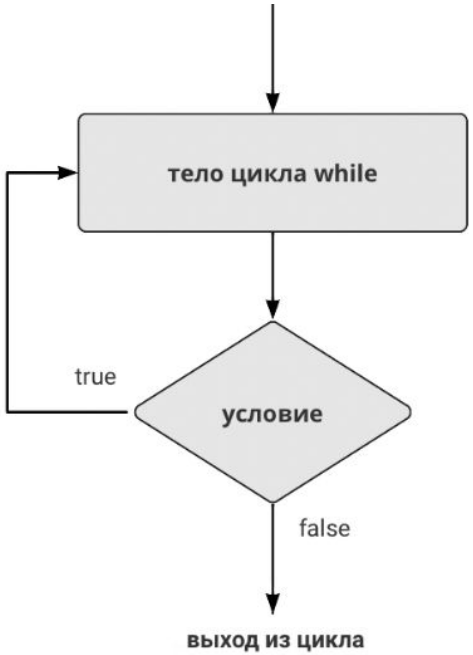
Цикл do...while (с постусловием)

Синтаксис

```
do {

// тело цикла

} while (условие)
```



Как это работает

1. Сначала выполняется тело цикла. Только затем оценивается условие.
2. Если условие — true, тело цикла выполняется снова.
3. Условие оценивается еще раз.
4. Если условие — true, тело цикла выполняется снова.
5. Этот процесс продолжается до тех пор, пока условие не станет false. Тогда цикл останавливается.

Сколько раз выполнится цикл?

```
const count = 1;
let i=0;
do {
   ++i;
}while(count == 1)
```

Сколько раз выполнится цикл?

```
let i=0;
while(true){
    i++
}
```

Сколько раз выполнится цикл?

```
let i = 0;

for (; i < 3;) {
  console.log( i++ );
}
```

```
let i = 0;
for (;;) {
  console.log( i++ );
}
```

Прерывание цикла: **break**

из цикла в любой момент можно выйти с помощью специальной директивы `break`, не дожидаясь условия. Директива `break` полностью прекращает выполнение цикла и передаёт управление на строку за телом цикла

Задача - подсчитывает сумму вводимых чисел до тех пор, пока посетитель их вводит

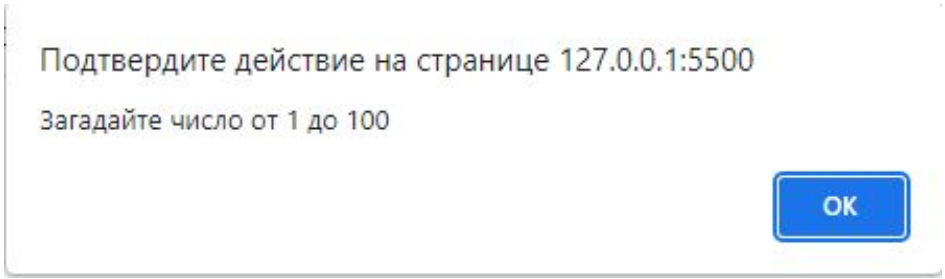
Переход к следующей итерации: **continue**

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.

Вывести только нечётные значения, используя **continue** :

Создание игры "Угадай число"



Создание игры "Камень, ножницы, бумага"

Игра состоит из трех раундов, каждый раз

- пользователь самостоятельно выбирает вариант (камень, ножницы, бумага),
- а компьютер генерирует случайное значение.

Условия выигрыша: камень затупляет ножницы, бумага накрывает камень, ножницы разрезают бумагу.

За каждую победу начисляется 1 балл. Выигрывает тот, кто сумел набрать больше очков.

Возможные исходы игры и каждого раунда: выиграл пользователь, выиграл компьютер, ничья.

Варианты

player_select	comp_select	WIN
К	Н	player_score + 1
Н	К	comp_score + 1
Н	Н	-
К	К	-

Все варианты сравнения выбора пользователя и выбора компьютера

Подсчёт итогового количества очков и определения победителя игры, с указанием количества побед у каждого игрока.

Реализуйте программу таким образом, чтобы пользователь мог повторить игру столько раз, сколько захочет