

Работа с по-сложни цикли

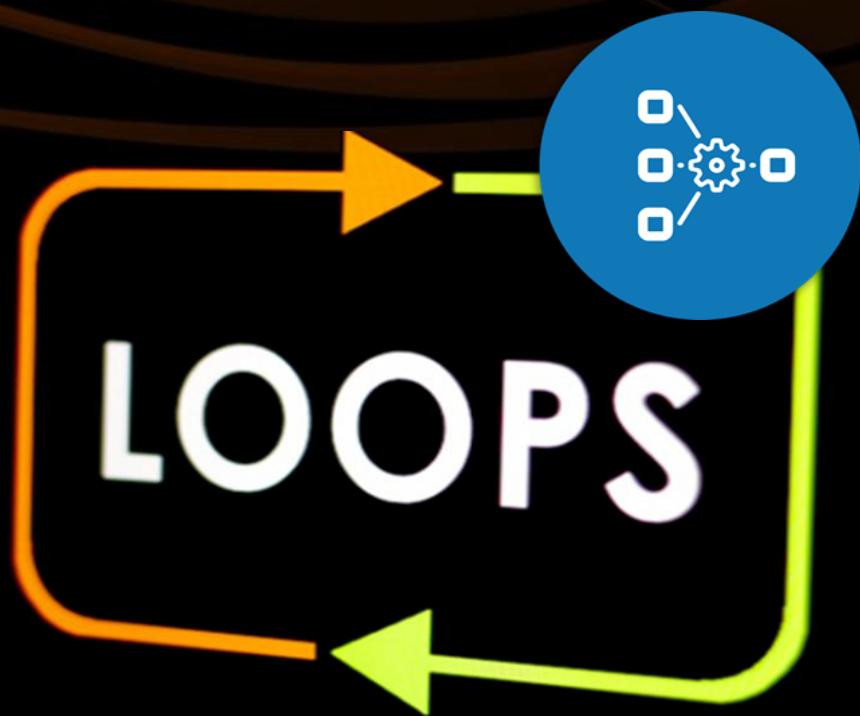


Софтуни
трейнърски екип
Софтуерен университет
<http://softuni.bg>

Цикли със стъпка, While, Do...While



По-сложни
цикли



Have a Question?



sli.do

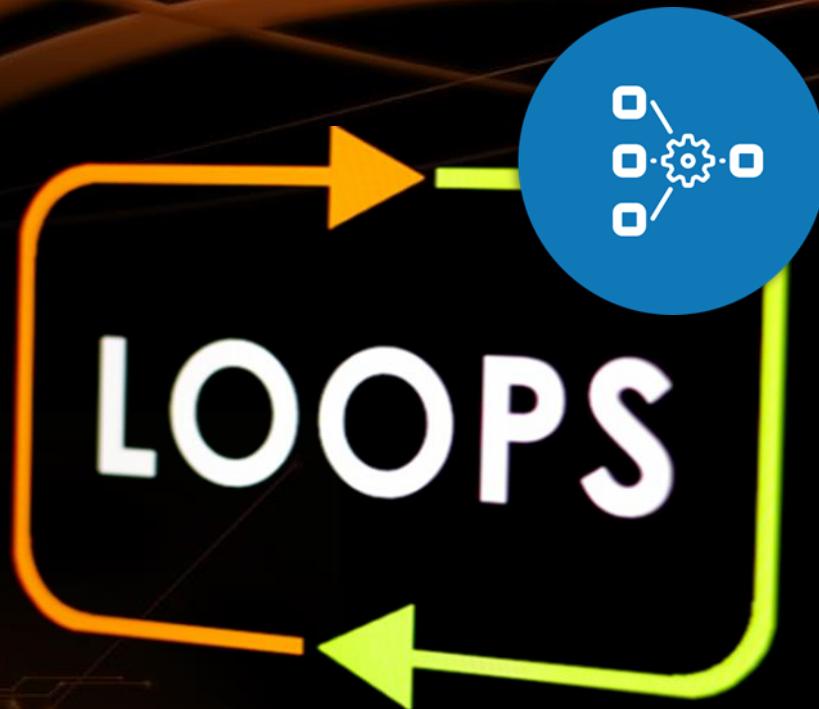
#TODO

Съдържание

■ По-сложни конструкции за цикъл:

- Цикъл със стъпка
- Цикъл с намаляваща стъпка
- Цикъл - **while**
- Цикъл - **do-while**
- Безкраен цикъл
 - Оператор **break**
 - Оператор **continue**





Цикли със стъпка

Работа с по-сложни `for`-цикли

Числата от 1 до N през 3 - условие

- Напишете програма, която:
 - Прочита цяло число **n**
 - Отпечатва числата от **1** до **n** със стъпка **3**
- Примерен вход и изход:

10  1, 4, 7, 10

15  1, 4, 7, 10, 13



Числата от 1 до N през 3 - решение

```
function loopByStep3([arg1]) {  
    let n = Number(arg1);  
    for (let i = 1; i <= n; i+=3) {  
        console.log(i);  
    }  
}
```

Задаване
на стъпка



Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#0>

Числата от N до 1 в обратен ред - условие

- Напишете програма, която:
 - Прочита цяло число **n**
 - Отпечатва числата от **n** до **1** в обратен ред (**стъпка -1**)
- Примерен вход и изход:

100 → 100, 99, 98, ..., 3, 2, 1

3 → 3, 2, 1

Числата от N до 1 в обратен ред - решение

```
function numbersNto1([arg1]) {  
    let n = Number(arg1);  
    for (let i = n; i >= 1; i-=1) {  
        console.log(i);  
    }  
}
```

Намаляваща
стъпка: -1

Обърнато
условие:
i >= 1



Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#1>

Числата от 1 до 2^n с for-цикъл – условие

- Напишете програма, която:
 - Прочита цяло число n
 - Отпечатва числата от 1 до 2^n
- Примерен вход и изход:

8	2	8	
16	128	4	2
2	4	2	4
2	32	8	

10  1, 2, 4, 8, 16, 32, ..., 1024

6  1, 2, 4, 8, 16, 32

Числата от 1 до 2^n с for-цикъл - решение

```
function powersOfTwo([arg1]) {  
    let n = Number(arg1);  
    let num = 1;  
    for (let i = 0; i <= n; i++) {  
        console.log(num);  
        num = num * 2;  
    }  
}
```

8	2	8	
16	128	4	2
2	4	2	4
2	32	8	

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#2>

Четни степени на 2 - условие

- Напишете програма, която:
 - Прочита цяло число **n**
 - Отпечатва четните степени на **2** до 2^n : $2^0, 2^2, 2^4, 2^8, \dots, 2^n$
- Примерен вход и изход:

10



1, 4, 16, ..., 1024

3



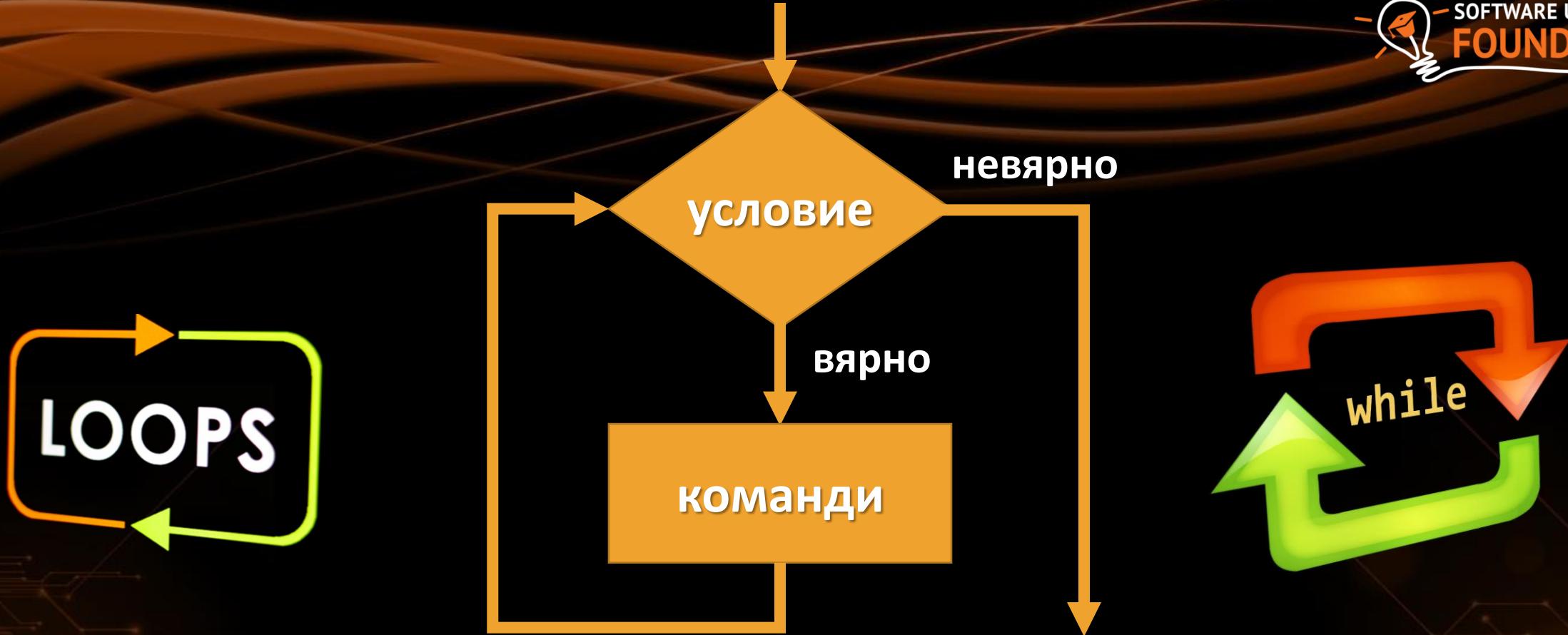
1, 4, 16

Четни степени на 2 - решение

```
function evenPowersOfTwo([arg1]) {  
    let n = Number(arg1);  
    let num = 1;  
    for (let i = 0; i <= n; i+=2) {  
        console.log(num);  
        num = num * 2 * 2;  
    }  
}
```

Ползваме
стъпка 2

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#3>



While цикъл

Повторение докато е в сила дадено условие

While цикъл

- Тялото на цикъла се изпълнява докато е вярно дадено условие

```
while (...) {  
    //code  
}
```

Условие (true/false)

Код за изпълнение (повторение)

Редица числа $2k+1$ - условие

- Напишете програма, която:
 - Прочита цяло число **n**
 - Отпечатва всички числа $\leq n$ от редицата: **1, 3, 7, 15, 31, ...**
 - Всяко следващо число е равно на предишното * 2 + 1

$1, (1*2)+1 = 3, (3*2)+1 = 7, (7*2)+1 = 15 \dots$

Редица числа $2k+1$ - решение

```
function sequence([arg1]) {  
    let n = Number(arg1);  
    let num = 1;  
    while (num <= n) {  
        console.log(num);  
        num = 2 * num + 1;  
    }  
}
```

Повтаряй докато е в
сила условието $num \leq n$

1, 3, 7, 15, 31, 63, ...

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#4>

Число в диапазона [1...100] - условие

- Напишете програма, която:
 - Прочита цяло число
 - Проверява дали е в диапазона [1...100]
 - При:
 - Намиране на число в диапазона,
прекратява изпълнение
 - Невалидно число прочита ново

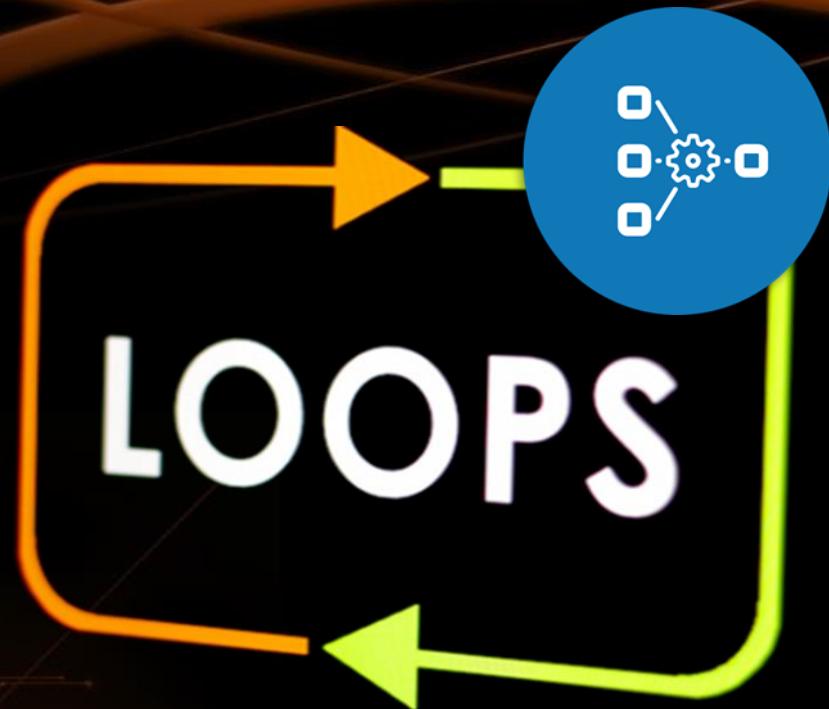


Число в диапазона [1...100] - решение

```
function numberInRange(args) {  
    let i = 0;  
    let num = Number(args[i]);  
    while (num < 1 || num > 100) {  
        i++;  
        console.log("Invalid number!");  
        num = Number(args[i]);  
    }  
    console.log(`The number is: ${num}`);  
}
```

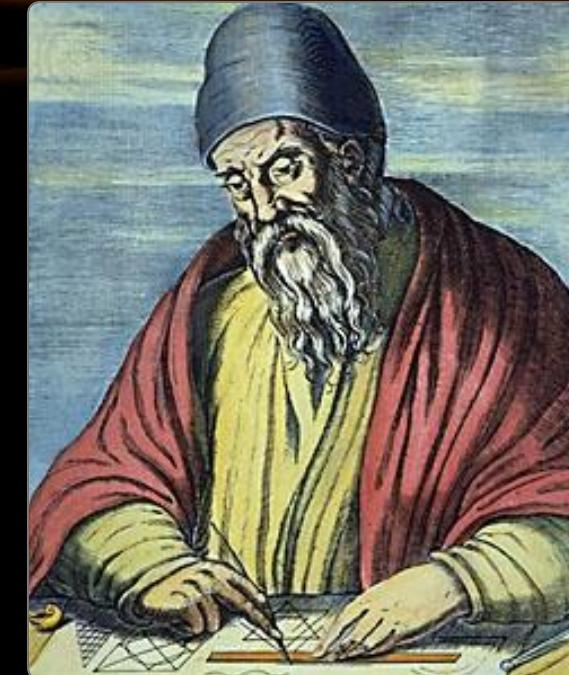
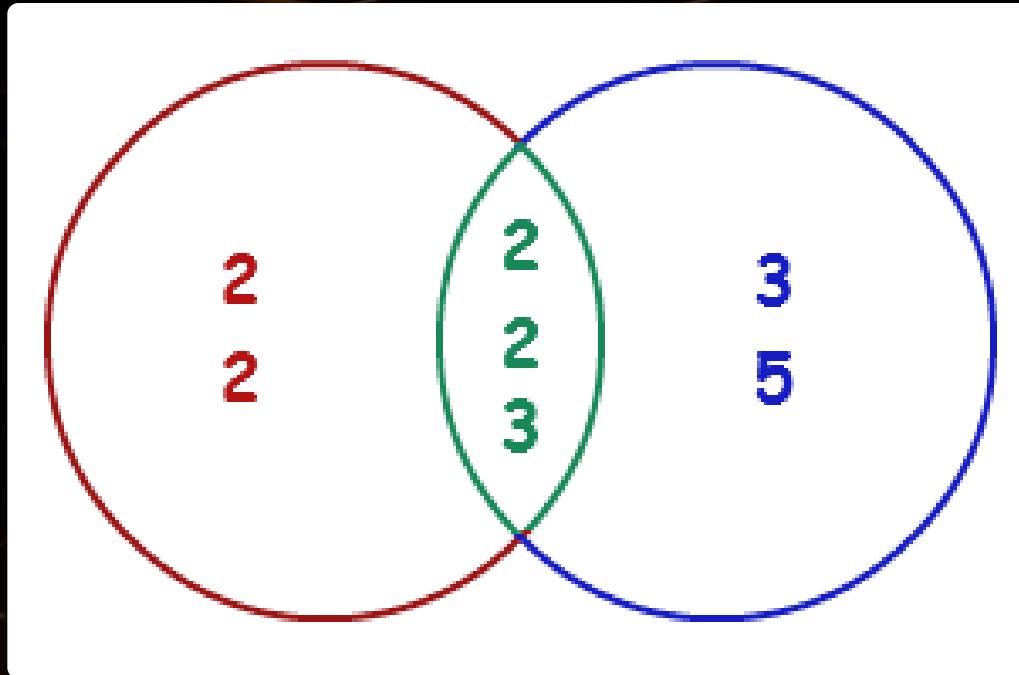


Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#5>



Цикли със стъпка и while цикъл

Работа на живо в клас (лаб)



Най-голям общ делител (НОД)

Алгоритъм на Евклид

Най-голям общ делител (НОД)

- Най-голям общ делител (НОД) на две естествени числа **a** и **b** е най-голямото число, което дели едновременно **a** и **b** без остатък
 - $\text{НОД}(24, 16) = 8$
 - $\text{НОД}(67, 18) = 1$
 - $\text{НОД}(12, 24) = 12$
 - $\text{НОД}(15, 9) = 3$
 - $\text{НОД}(10, 10) = 10$
 - $\text{НОД}(100, 88) = 4$



Алгоритъм на Евклид за НОД - условие

- Напишете програма, която:
 - Прочита 2 цели числа **a** и **b**
 - Намира най-големия им общ делител - **НОД(a, b)**
- Насоки:
 - Докато не се достигне остатък 0:
 - Дели се по-голямото число на по-малкото
 - Взема се остатъка от делението

```
while b ≠ 0
    var oldB = b;
    b = a % b;
    a = oldB;
print a;
```

Алгоритъм на Евклид за НОД - решение



```
function gcd([arg1, arg2]) {  
    let a = Number(arg1);  
    let b = Number(arg2);  
    while (b != 0) {  
        let oldB = b;  
        b = a % b;  
        a = oldB;  
    }  
    console.log("GCD = " + a);  
}
```

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#6>



Do...While цикъл

Повторение докато е изпълнено условието

Do-while цикъл

- Тялото на цикъла се изпълнява докато е вярно дадено условие
 - Изпълнява се минимум един път

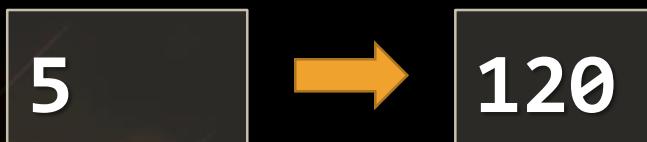
```
do {  
    //code  
} while (...);
```

Код за
изпълнение
(повторение)

Условие (true/false)

Изчисляване на факториел - условие

- Напишете програма, която:
 - Прочита естествено число **n**
 - Изчислява факториел от n ($n!$)
- Примерен вход и изход:
 - $5! = 1 * 2 * 3 * 4 * 5 = 120$



$n!$

Изчисляване на факториел - решение



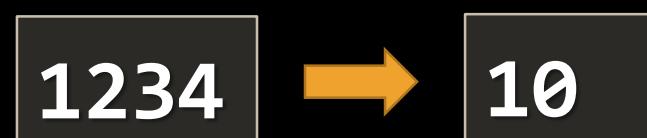
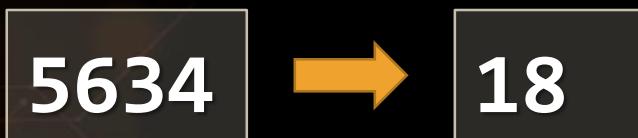
```
function factorial([arg1]) {  
    let n = Number(arg1);  
    let fact = 1;  
    do {  
        fact = fact * n;  
        n--;  
    } while (n > 1);  
    console.log(fact);  
}
```

n!

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#7>

Сумиране на цифрите на число - условие

- Напишете програма, която:
 - Прочита цяло положително число **n**
 - Сумира цифрите на **n**
- Примерен вход и изход:
 - $n = 5634: 5 + 6 + 3 + 4 = 18$



Сумиране на цифрите на число - решение

```
function sumDigits([arg1]) {  
    let n = Number(arg1);  
    let sum = 0;  
    do {  
        sum = sum + (n % 10);  
        n = Math.floor(n / 10);  
    } while (n > 0);  
    console.log("Sum of digits: " + sum);  
}
```

n % 10 връща последната цифра на числото n

Math.floor(n / 10) изтрива последната цифра на n

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#8>



Безкрайни цикли и оператори *break* и *continue*

Безкраен цикъл

- Безкраен цикъл имаме когато:
 - Нямаме условие, което да прекрати цикъла
 - Нямаме команда, която да прекрати цикъла

```
while(true){  
    console.log("Infinite loop");  
}
```



```
for ( ; ; ){  
    console.log("Infinite loop");  
}
```



Условия за прекратяване на цикъл

```
while(...)  
{  
    console.log("Infinite loop");  
}
```

Условие за
прекратяване на
цикъл



```
for (;...;)  
{  
    console.log("Infinite loop");  
}
```

Условие за
прекратяване на
цикъл



Команда за прекратяване на цикъл

- Оператор **break** – прекъсване на цикъла

```
while(true)
{
    console.log("Infinite loop");
    if (...){
        break;
    }
}
```

Условие за прекратяване на цикъл

Команда за излизане от цикъл



Проверка за просто число - условие

- Напишете програма, която:
 - Прочита цяло число **n**
 - Проверява да ли **n** е просто число
- Насоки:
 - Едно число **n** е **просто**, ако се дели единствено на **1** и **n** и е поголямо от **1**
 - Прости числа: **2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, ...**
 - Непрости (композитни) числа: $10 = 2 * 5, 21 = 3 * 7, 143 = 13 * 11$

Проверка за просто число - решение

```
function isPrime([arg1]) {  
    let n = Number(arg1);  
    let prime = true;  
    for (let i = 2; i <= Math.sqrt(n); i++)  
        if (n % i == 0) {  
            prime = false;  
            break; // break излиза от цикъла  
        }  
    if (prime && n > 2) console.log("Prime");  
    else console.log("Not prime");  
}
```

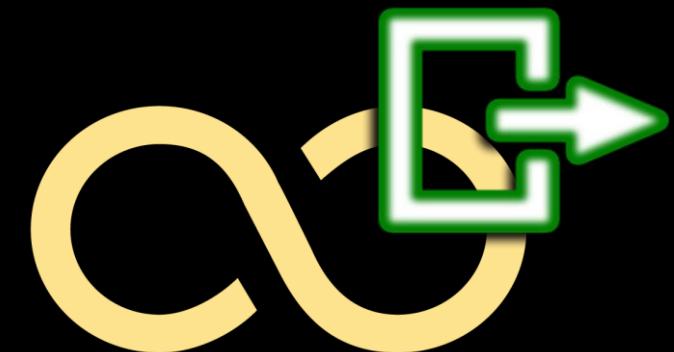
break излиза от цикъла



Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#9>

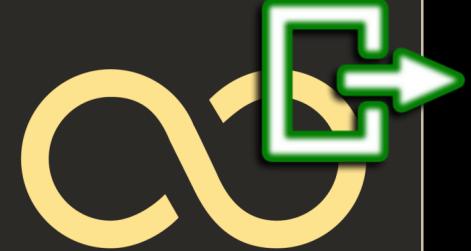
Четно число - условие

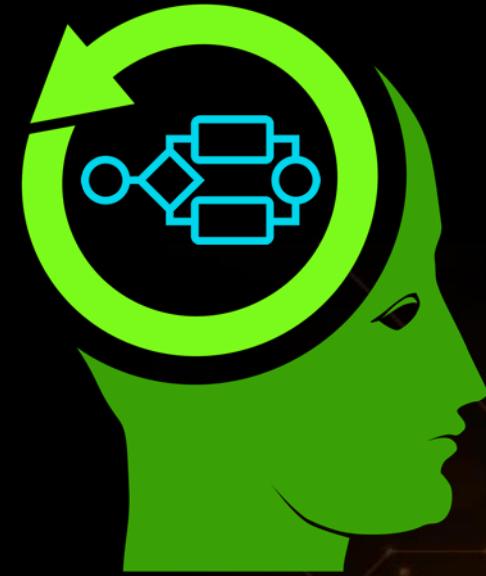
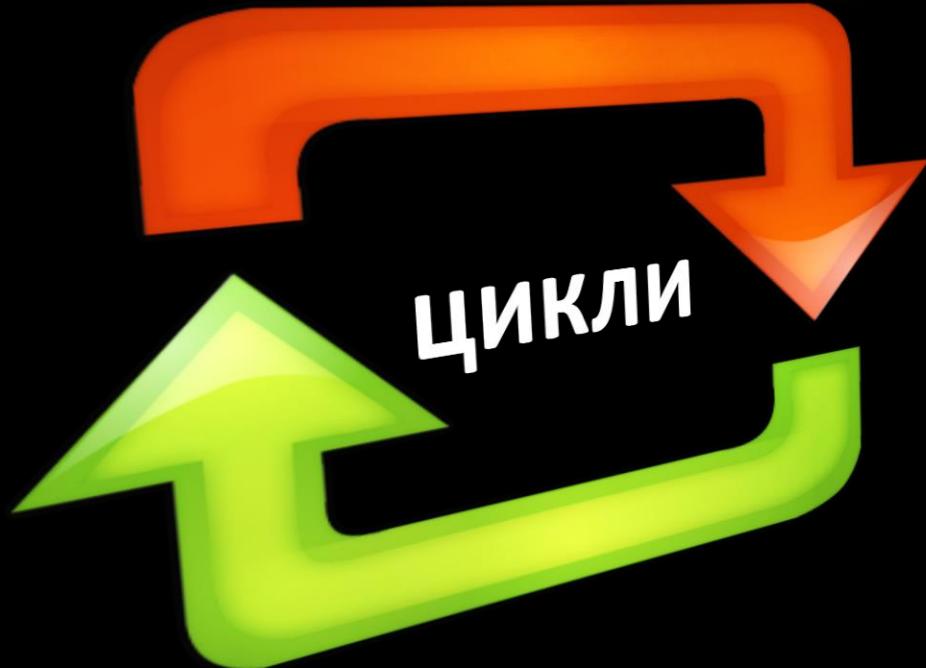
- Напишете програма, която:
 - Прочита число **n**
 - Проверява дали **n** е четно
 - При невалидно число се връща към повторно въвеждане



Четное число - решение

```
function enterEvenNumber(args) {  
    let i = 0;  
    let num = 0;  
    while (true) {  
        num = Number(args[i]);  
        if (num % 2 == 0)  
            break; // even number -> exit from the loop  
        console.log("The number is not even.");  
        i++;  
    }  
    console.log(`Even number entered: ${num}`);  
}
```

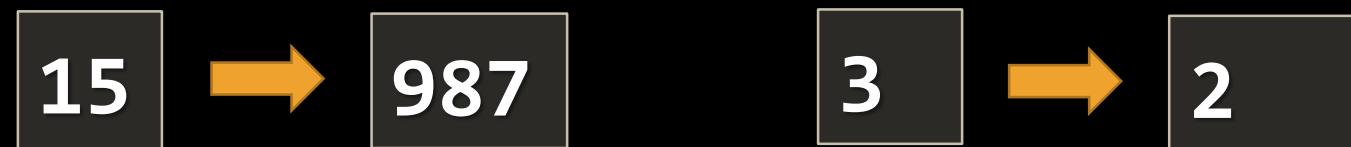




Задачи с цикли

Числа на Фиbonачи - условие

- Напишете програма, която:
 - Прочита число **n**
 - Пресмята **n-тото** число на Фиbonачи
- Числата на Фиbonачи са следните: **1, 1, 2, 3, 5, 8, 13, 21, 34, ...**
 - $F_0 = 1$
 - $F_1 = 1$
 - $F_n = F_{n-1} + F_{n-2}$
 - Примерен вход и изход:
 - $F(15) = 987$



Числа на Фиbonачи - решение

```
function fibonacci([arg1]) {  
    let n = Number(arg1);  
    let f0 = 1;  
    let f1 = 1;  
    for (let i = 0; i < n-1; i++) {  
        let fNext = f0 + f1;  
        f0 = f1;  
        f1 = fNext;  
    }  
    console.log(f1);  
}
```



Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#11>

Пирамида от числа – условие

- Да се отпечатат числата 1...n в пирамида като в примерите:

n = 7



1
2 3
4 5 6
7

n = 10



1
2 3
4 5 6
7 8 9 10

n = 12



1
2 3
4 5 6
7 8 9 10
11 12

n = 15



1
2 3
4 5 6
7 8 9 10
11 12 13 14 15



Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#12>

Пирамида от числа – решение

```
function numberPyramid([arg1]) {  
    let n = Number(arg1);  
    let num = 1;  
    let result = "";  
    for (let row = 1; row <= n; row++) {  
        for (let col = 1; col <= row; col++) {  
            if (col > 1) result += " ";  
            result += num;  
            num++;  
            if (num > n) break;  
        }  
        console.log(result);  
        result = "";  
        if (num > n) break;  
    }  
}
```

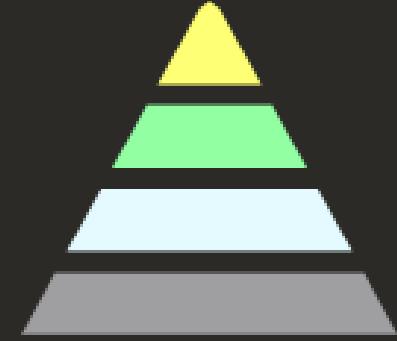


Таблица с числа - условие

- Да се отпечатат числата $1 \dots n$ в таблица като в примерите:

$n = 2$



1	2
2	1



$n = 3$



1	2	3
2	3	2
3	2	1

$n = 4$



1	2	3	4
2	3	4	3
3	4	3	2
4	3	2	1

$n = 5$



1	2	3	4	5
2	3	4	5	4
3	4	5	4	3
4	5	4	3	2
5	4	3	2	1

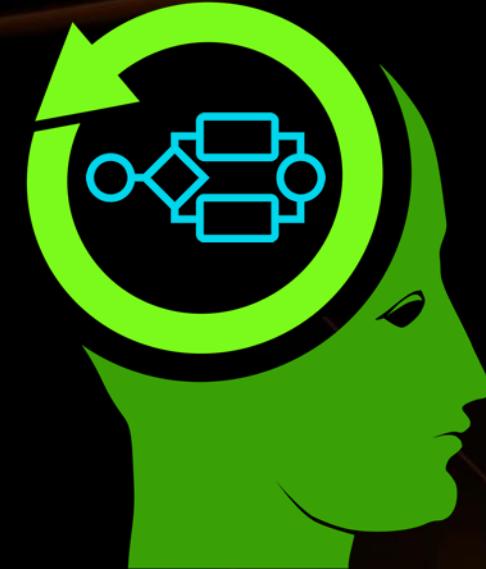
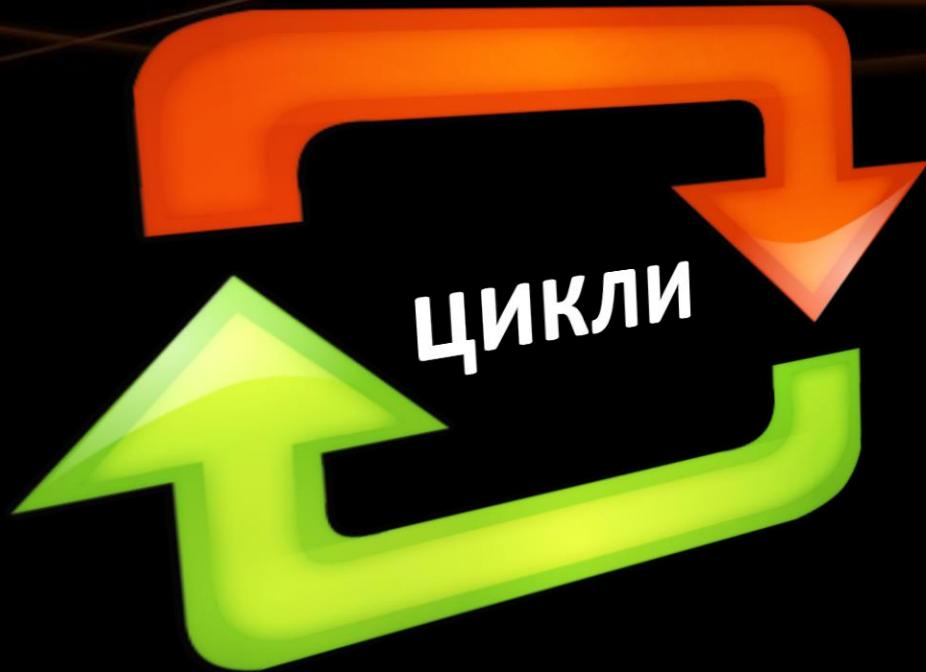
Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#13>

Таблица с числа – решение

```
function numberTable([arg1]) {  
    let n = Number(arg1);  
    let result = "";  
    for (let row = 0; row < n; row++) {  
        for (let col = 0; col < n; col++) {  
            let num = row + col + 1;  
            if (num > n) num = 2 * n - num;  
            result = result + num + " ";  
        }  
        console.log(result);  
        result = "";  
    }  
}
```

1	2	3	4	5
2	3	4	5	4
3	4	5	4	3
4	5	4	3	2
5	4	3	2	1

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#13>



По-сложни задачи с цикли

Работа на живо в клас (лаб)

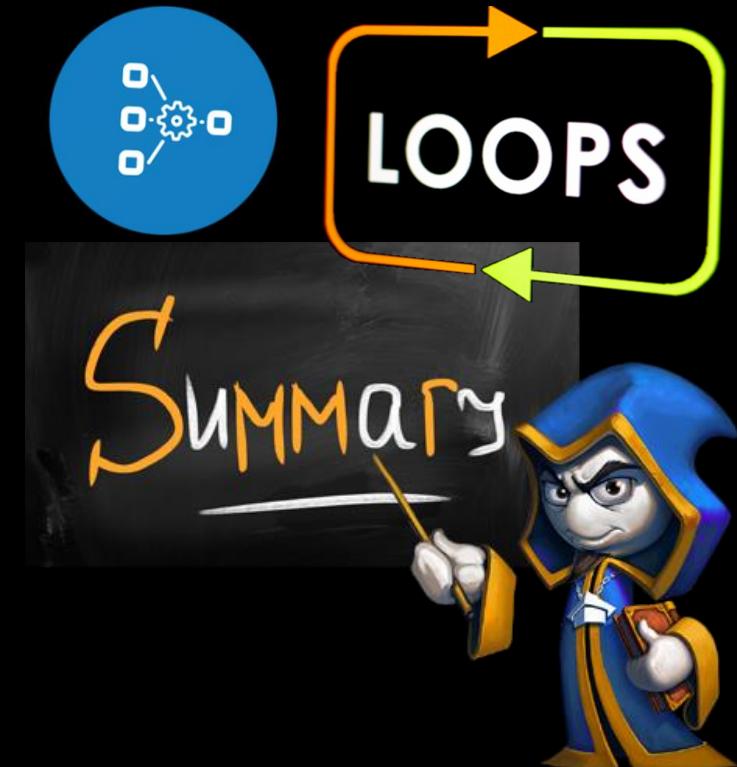
Какво научихме днес?

- Можем да ползваме **for**-цикли със стъпка:

```
for (let i = 1; i <= n; i+=3)  
    console.log(i);
```

- Цикли **while** / **do-while** повтаря докато е в сила дадено условие:

```
let num = 1;  
while (num <= n)  
    console.log(num++);
```

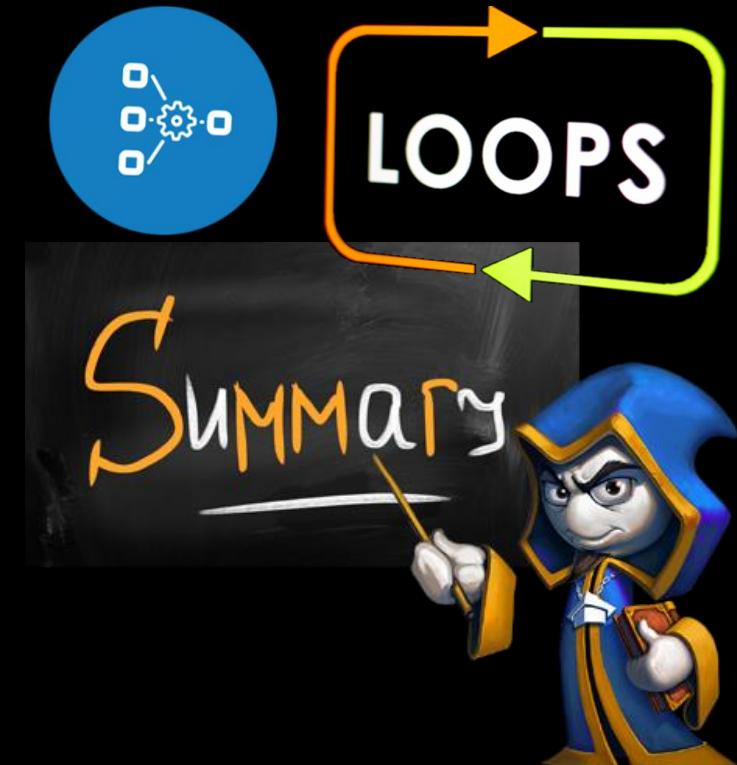


Какво научихме днес? (2)

- Можем да създаваме безкрайни цикли и когато се наложи да излизаме от тях:

```
for (;;) {  
    if (...)  
        break;  
}
```

```
while (true){  
    if (...)  
        break;  
}
```



Чертане с цикли



Въпроси?

SUPERHOSTING.BG

INDEAVR
Serving the high achievers



Лиценз

- Настоящият курс (слайдове, примери, видео, задачи и др.) се разпространяват под свободен лиценз "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International"



Бесплатни обучения в СофтУни

- Фондация "Софтуерен университет" – softuni.org
- Софтуерен университет – качествено образование, професия и работа за софтуерни инженери
 - softuni.bg
- СофтУни @ Facebook
 - facebook.com/SoftwareUniversity
- СофтУни форуми – forum.softuni.bg

