

Цикли

Цикл — різновид керівної конструкції у високорівневих мовах програмування, призначений для організації багаторазового виконання набору інструкцій (команд).

Ти ж програміст ...



<https://reactor.cc/post/1278775>

[https://uk.wikipedia.org/wiki/%D0%A6%D0%B8%D0%BA%D0%BB_\(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F\)](https://uk.wikipedia.org/wiki/%D0%A6%D0%B8%D0%BA%D0%BB_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F))

Для чого потрібні цикли?

```
//Вивести 1 раз привітання "Hello"  
  
document.write('Hello')
```

Для чого потрібні цикли?

```
//Вивести 2 рази привітання "Hello"
```

```
document.write('Hello')
```

```
document.write('Hello')
```

Для чого потрібні цикли?

```
//Вивести 3 рази привітання "Hello"  
  
document.write('Hello')  
document.write('Hello')  
document.write('Hello')
```

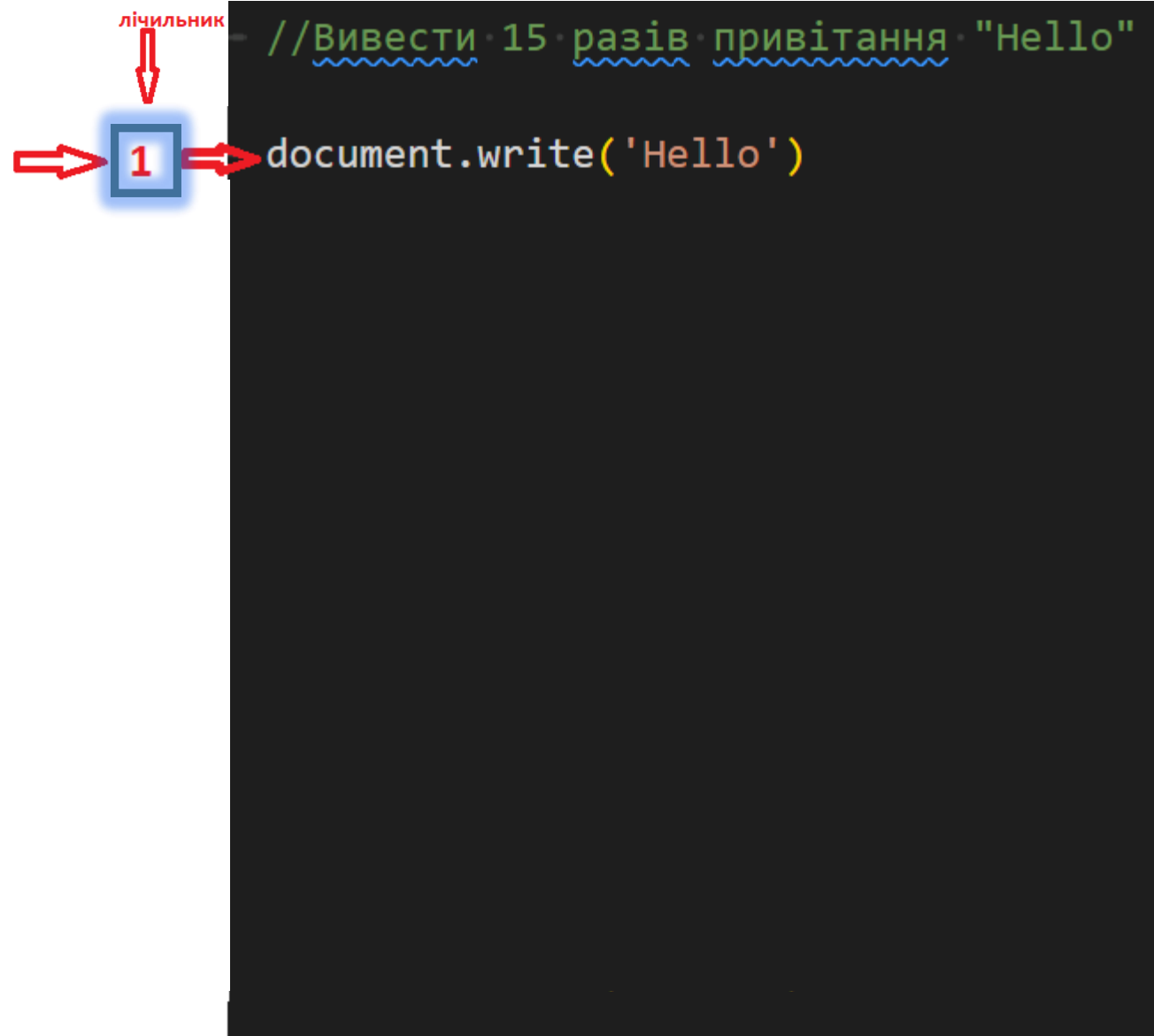
Для чого потрібні цикли?

```
//Вивести 15 разів привітання "Hello"
```

```
document.write('Hello')  
document.write('Hello')  
document.write('Hello')  
document.write('Hello')  
document.write('Hello')  
document.write('Hello')  
document.write('Hello')  
document.write('Hello')  
document.write('Hello')  
document.write('Hello')  
document.write('Hello')  
document.write('Hello')  
document.write('Hello')  
document.write('Hello')
```

15

Для чого потрібні цикли?



Для чого потрібні цикли?

лічильник



1



```
document.write('Hello')
```

2



```
document.write('Hello')
```

```
//Вивести 15 разів привітання "Hello"
```


Для чого потрібні цикли?

лічильник



```
//Вивести 15 разів привітання "Hello"
```

1 ⇒ document.write('Hello')

2 ⇒ document.write('Hello')

3 ⇒ document.write('Hello')



Для чого потрібні цикли?

лічильник



```
//Вивести 15 разів привітання "Hello"
```

1 ⇒ document.write('Hello')

2 ⇒ document.write('Hello')

3 ⇒ document.write('Hello')

4 ⇒ document.write('Hello')



Для чого потрібні цикли?

лічильник

↓
//Вивести 15 разів привітання "Hello"

1 ⇒ document.write('Hello')

2 ⇒ document.write('Hello')

3 ⇒ document.write('Hello')

4 ⇒ document.write('Hello')

5 ⇒ document.write('Hello')

.. ⇒ document.write('Hello')

.. ⇒ document.write('Hello')

.. ⇒ document.write('Hello')

.. ⇒ document.write('Hello')

.. ⇒ document.write('Hello')

.. ⇒ document.write('Hello')

.. ⇒ document.write('Hello')

13 ⇒ document.write('Hello')

14 ⇒ document.write('Hello')

⇒ 15 ⇒ document.write('Hello')

15

Для чого потрібні цикли?

початкове значення

лічильник

```
//Вивести 15 разів привітання "Hello"
```

1 document.write('Hello')

2 document.write('Hello')

3 document.write('Hello')

4 document.write('Hello')

5 document.write('Hello')

.. document.write('Hello')

.. document.write('Hello')

.. document.write('Hello')

.. document.write('Hello')

.. document.write('Hello')

.. document.write('Hello')

13 document.write('Hello')

14 document.write('Hello')

15 document.write('Hello')

кінцеве значення

15

Цикли

Часто постає потреба виконати один і той самий оператор декілька разів. для цього застосовують *оператори циклів*. Цикл складається із *заголовка* і *тіла*. У заголовку циклу зазначається умова завершення циклу, а тіло циклу являє собою оператор, який потрібно виконати декілька разів. Кожне виконання оператора тіла циклу називається його *ітерацією*.

У JavaScript, як і в більшості інших мов програмування, є три типи циклів: з передумовою, післяумовою та параметром.

Цикл з параметром `for`

Найчастіше у випадку, коли повторення потрібно робити у залежності від зміни деякого параметра або ж у випадку, коли потрібно повторювати тіло циклу деяку кількість разів використовують цикл з параметром `for`.

Загальний вигляд:

```
for (<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)  
    <оператор>;
```

Схематичне зображення виконання оператора:

```
for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)  
    <оператор>;
```

Схематичне зображення виконання оператора:

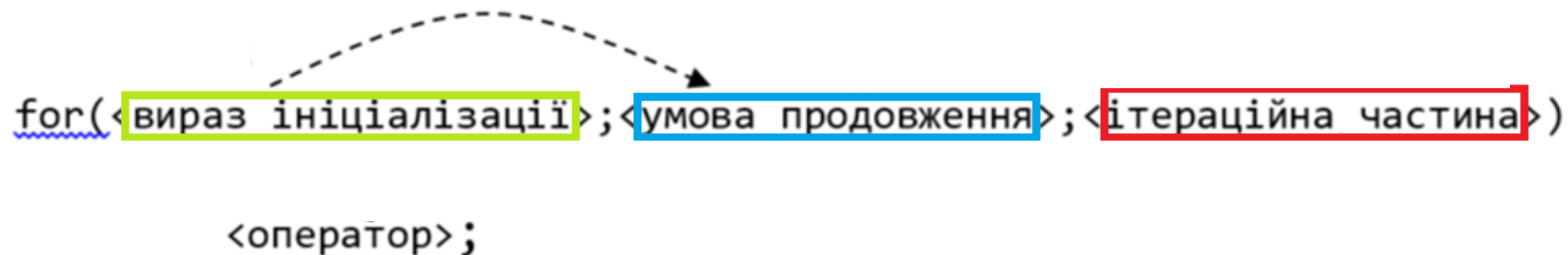
Початок



```
for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)
```

```
    <оператор>;
```

Схематичне зображення виконання оператора:

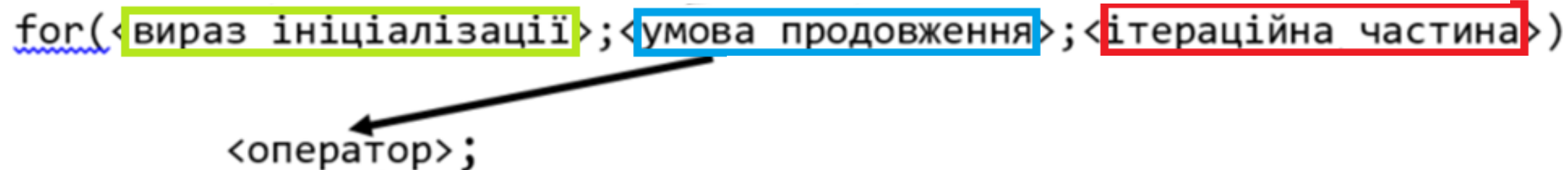


The diagram illustrates the components of a for loop. The word `for` is underlined in blue. The opening parenthesis `(` is followed by three components separated by semicolons: `<вираз ініціалізації>` (highlighted with a green box), `<умова продовження>` (highlighted with a blue box), and `<ітераційна частина>` (highlighted with a red box). A dashed curved arrow originates from the first component and points to the second, indicating the flow of execution. The closing parenthesis `)` is followed by `<оператор>;`.

```
for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)  
    <оператор>;
```


Схематичне зображення виконання оператора:

```
for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)  
    <оператор>;
```



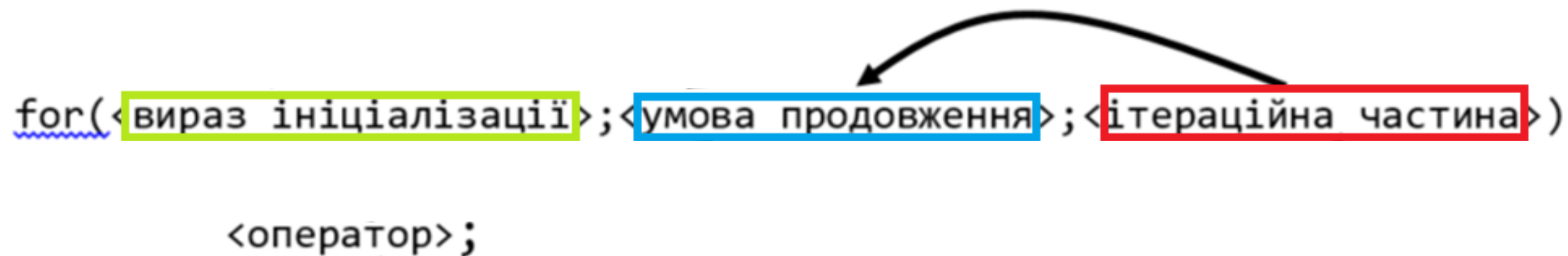
Схематичне зображення виконання оператора:

for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)

<оператор>;



Схематичне зображення виконання оператора:

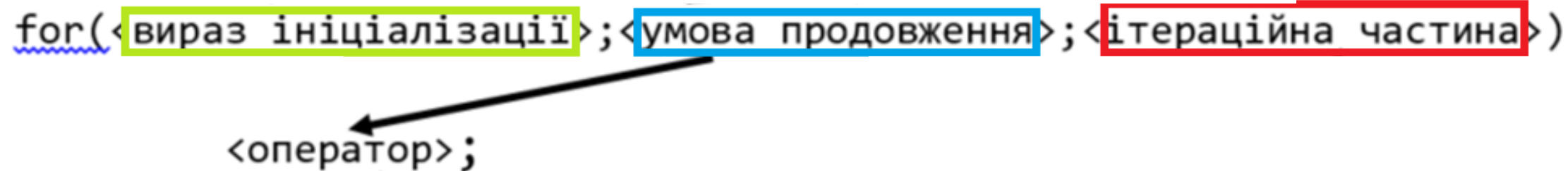


The diagram illustrates the components of a for loop. The loop header is enclosed in parentheses and contains three parts: an initialization expression, a continuation condition, and an iteration body. These parts are highlighted with colored boxes: a yellow box for the initialization expression, a blue box for the continuation condition, and a red box for the iteration body. A curved arrow points from the iteration body back to the continuation condition, indicating the loop's flow. The iteration body is followed by a semicolon and the keyword 'operator'.

```
for( <вираз ініціалізації>; <умова продовження>; <ітераційна частина> )  
    <оператор>;
```

Схематичне зображення виконання оператора:

```
for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)  
    <оператор>;
```



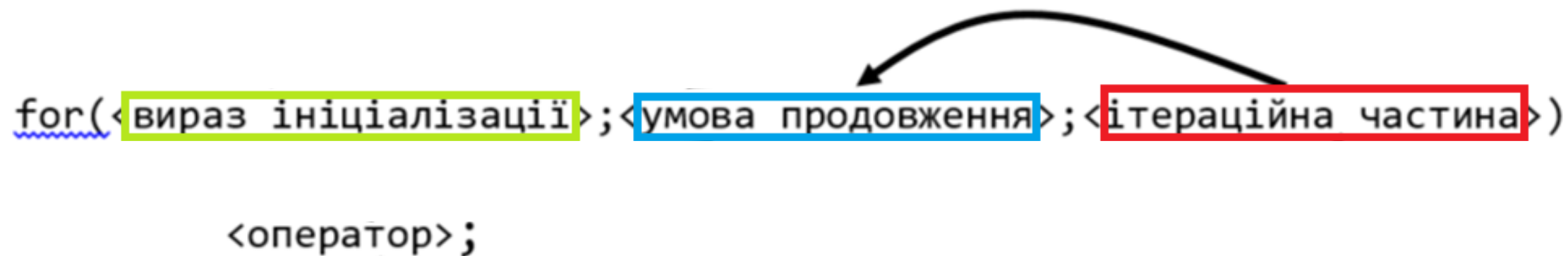
Схематичне зображення виконання оператора:

for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)

<оператор>;



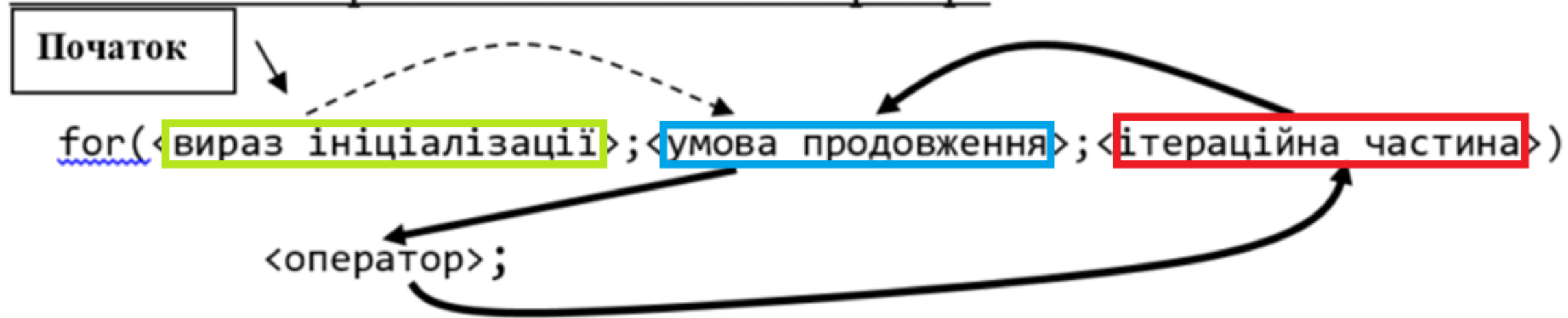
Схематичне зображення виконання оператора:



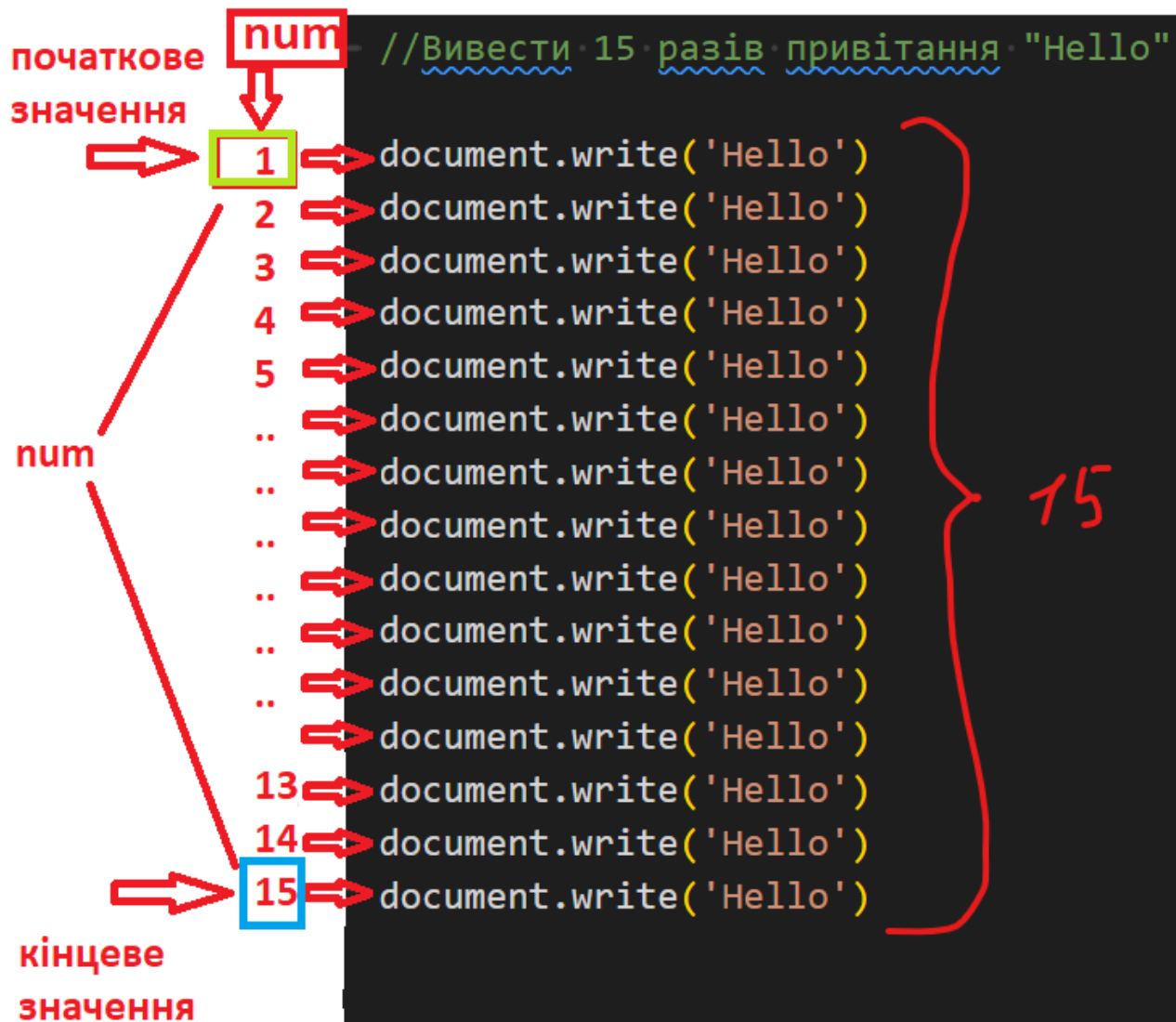
The diagram illustrates the execution of a for loop. The loop header is shown as `for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)`. The components are highlighted with colored boxes: a yellow box for the initialization expression, a blue box for the continuation condition, and a red box for the iteration part. A curved arrow points from the iteration part back to the continuation condition, indicating the loop's flow. Below the header, the body of the loop is represented by `<оператор>;`.

```
for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)  
    <оператор>;
```

Схематичне зображення виконання оператора:

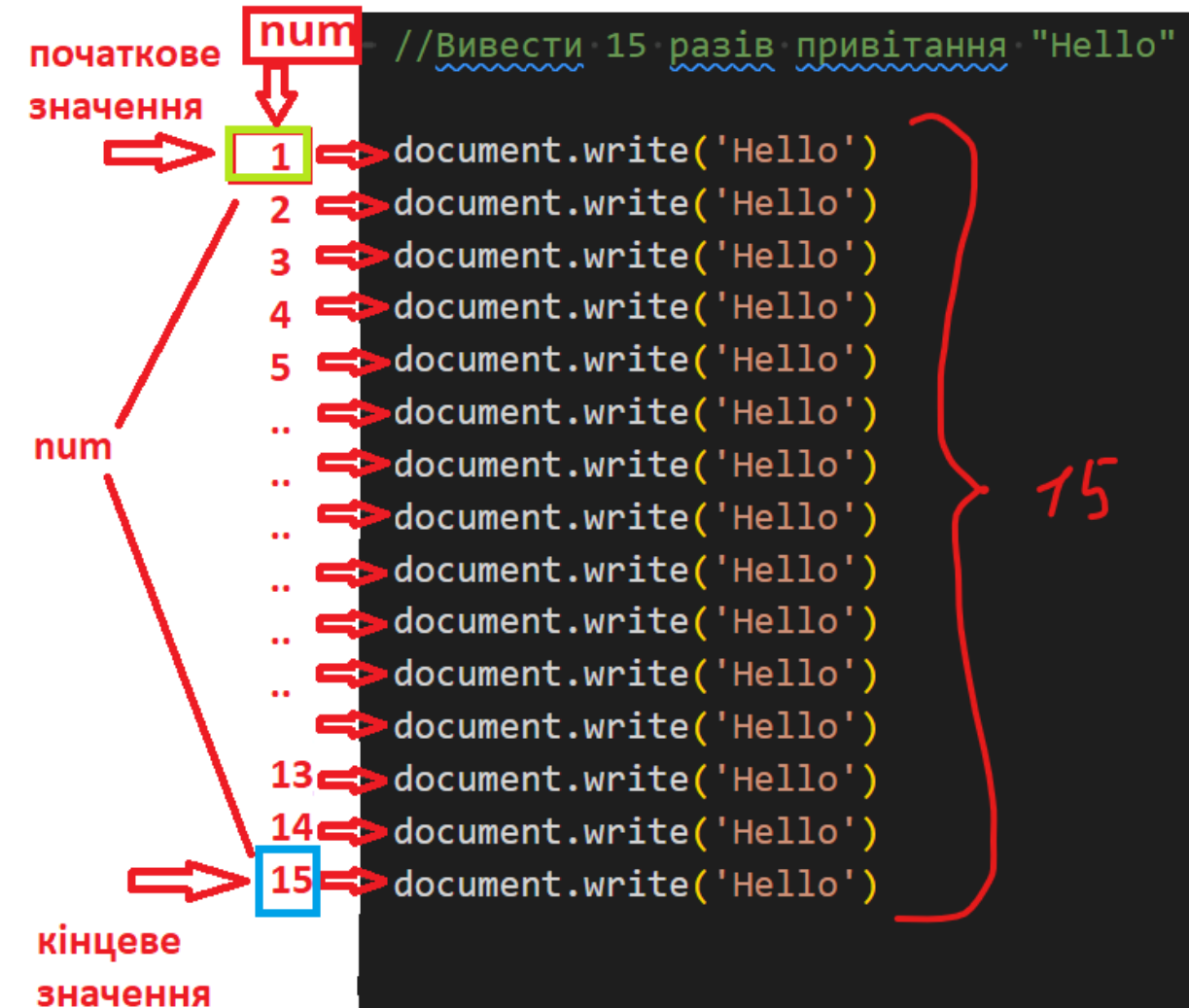


Для чого потрібні цикли?



Для чого потрібні цикли?

```
for (параметр = поч.значення; параметр <= кін.значення; параметр = параметр + крок)
{
    оператор ;
}
```



Для чого потрібні цикли?

```
for ( параметр = поч.значення; параметр <= кін.значення; параметр = параметр + крок )  
{  
    оператор ;  
}
```

початкове значення

num

```
//Вивести 15 разів привітання "Hello"  
1 document.write('Hello')  
2 document.write('Hello')  
3 document.write('Hello')  
4 document.write('Hello')  
5 document.write('Hello')  
.. document.write('Hello')  
.. document.write('Hello')  
.. document.write('Hello')  
.. document.write('Hello')  
.. document.write('Hello')  
.. document.write('Hello')  
13 document.write('Hello')  
14 document.write('Hello')  
15 document.write('Hello')
```

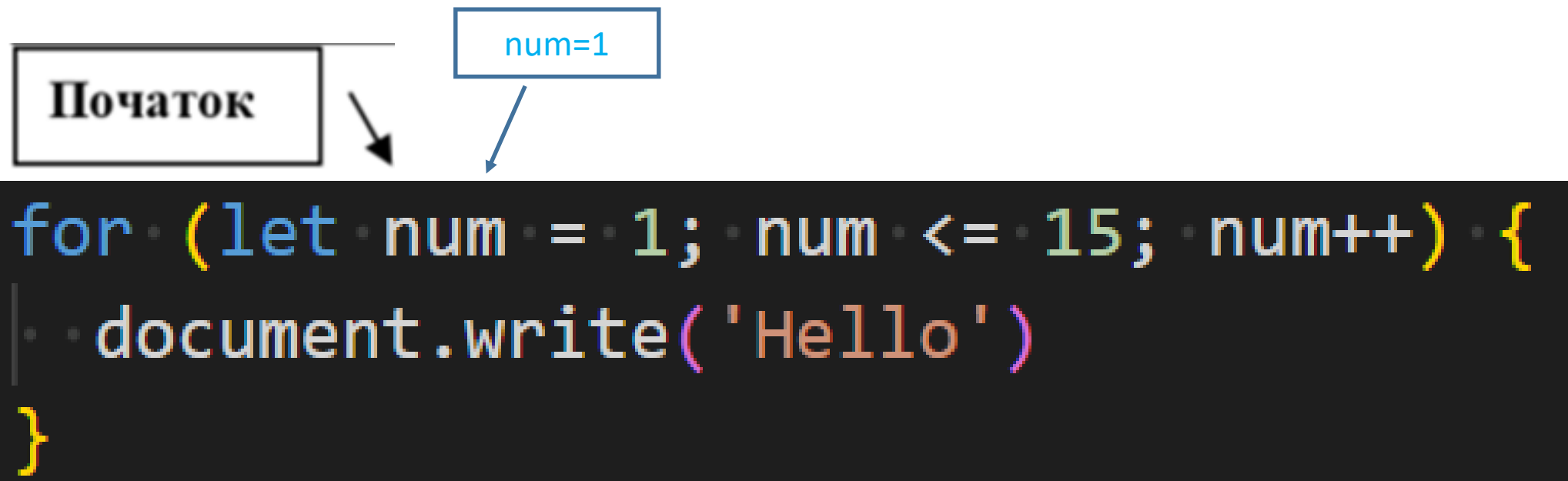
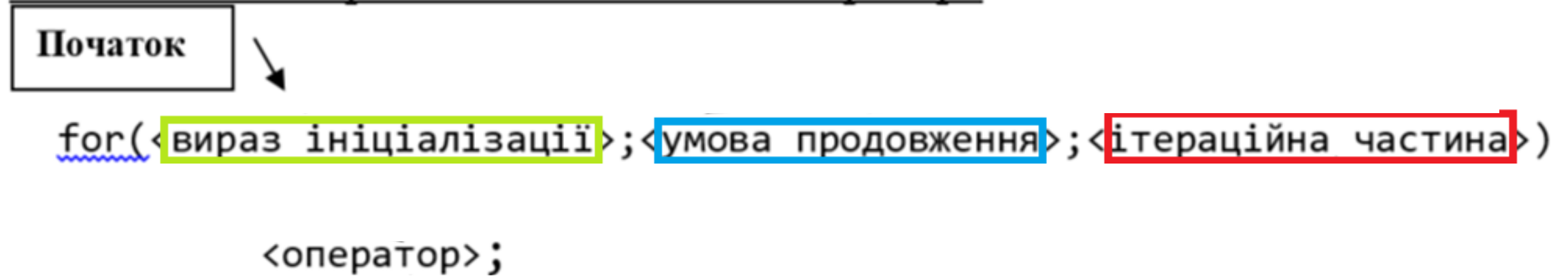
num

кінцеве значення

15

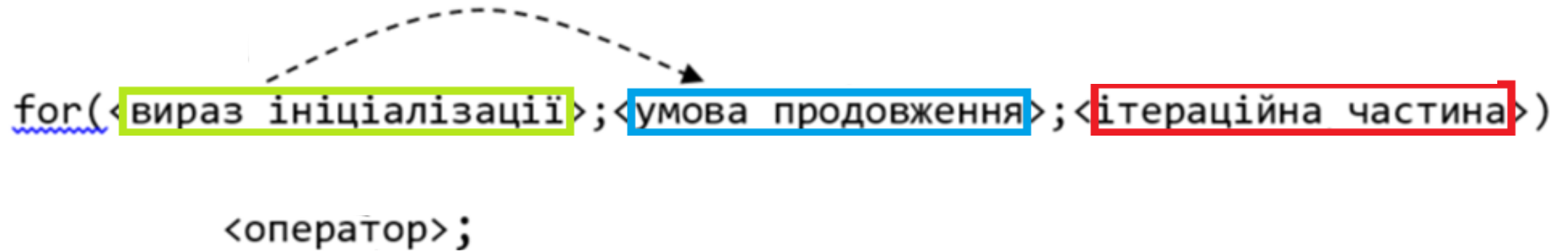
```
for (let num = 1; num <= 15; num++) {  
    document.write('Hello')  
}
```

Схематичне зображення виконання оператора:

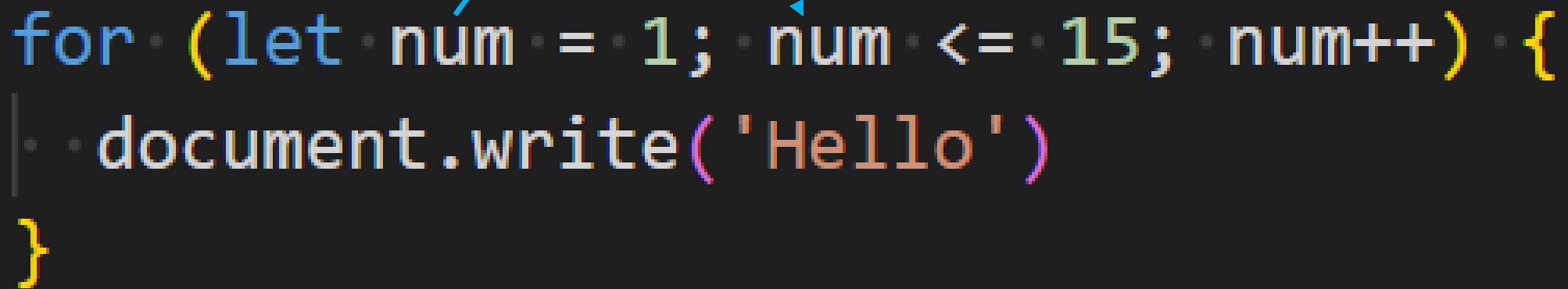


Схематичне зображення виконання оператора:

for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)
 <оператор>;

A diagram illustrating the components of a for loop. The text 'for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)' is shown with three colored boxes: a green box around '<вираз ініціалізації>', a blue box around '<умова продовження>', and a red box around '<ітераційна частина>'. A dashed black arrow points from the green box to the blue box. Below this, the text '<оператор>;' is shown.

for (let num = 1; num <= 15; num++) {
 document.write('Hello')
}

A diagram showing the execution of a JavaScript for loop. The code 'for (let num = 1; num <= 15; num++) { document.write('Hello') }' is shown in a dark box. A dashed blue arrow points from the 'num = 1' part of the code to a box labeled 'num=1'. Another dashed blue arrow points from the 'num++' part of the code to the same 'num=1' box. A solid blue arrow points from the 'num=1' box to the 'num <= 15' part of the code.

Схематичне зображення виконання оператора:

for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)

<оператор>;

num=1

```
for (let num = 1; num <= 15; num++) {  
  document.write('Hello')  
}
```

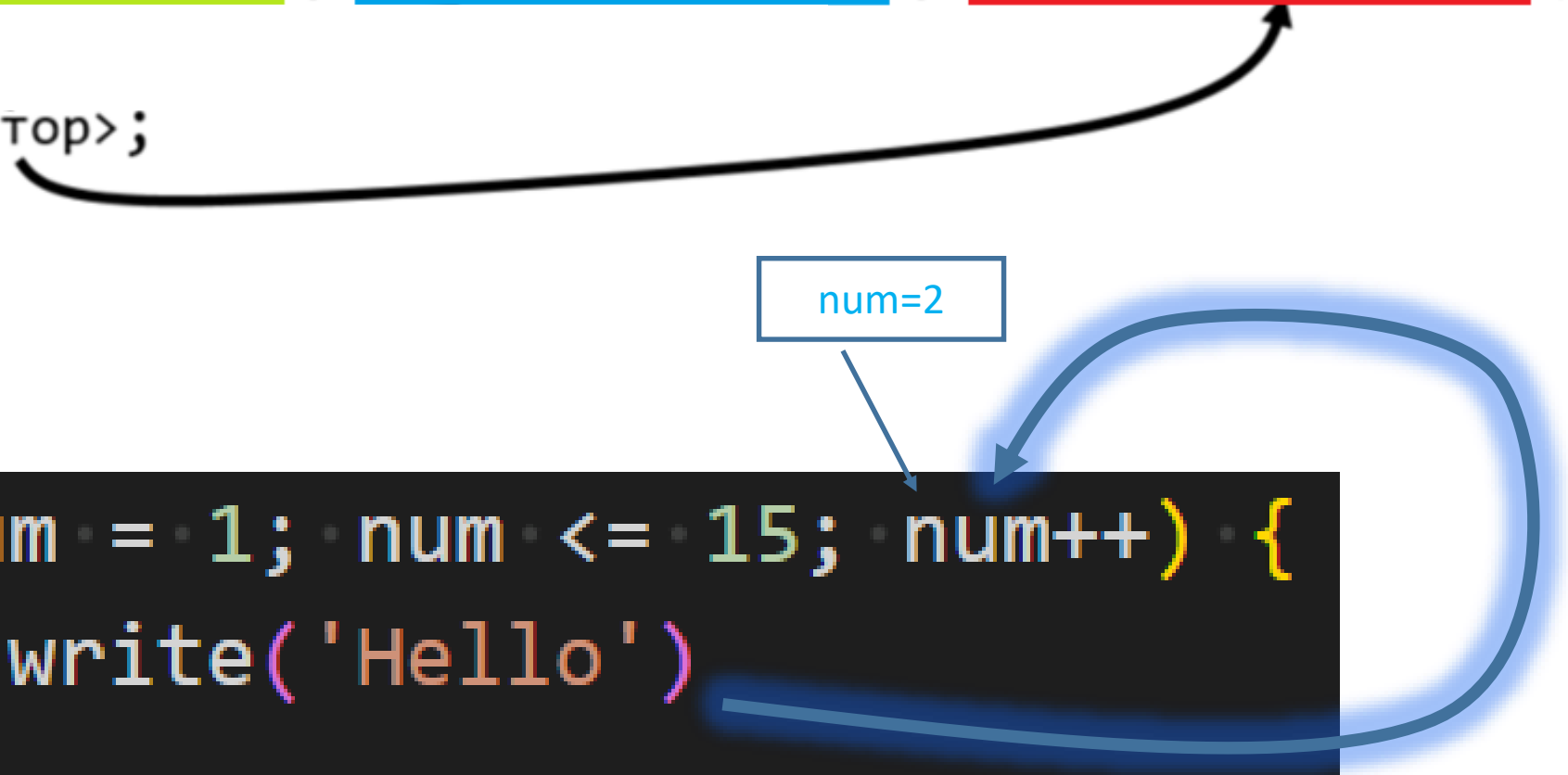
Схематичне зображення виконання оператора:

for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)

<оператор>;


```
for (let num = 1; num <= 15; num++) {  
  document.write('Hello')  
}
```

num=2

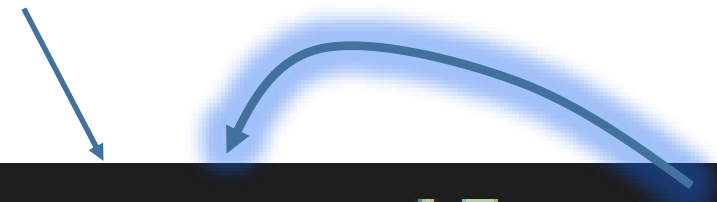


Схематичне зображення виконання оператора:

for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)
 <оператор>;



num=2



```
for (let num = 1; num <= 15; num++) {  
  document.write('Hello')  
}
```

Схематичне зображення виконання оператора:

for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)

<оператор>;

num=2

```
for (let num = 1; num <= 15; num++) {  
  document.write('Hello')  
}
```


Схематичне зображення виконання оператора:

for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)

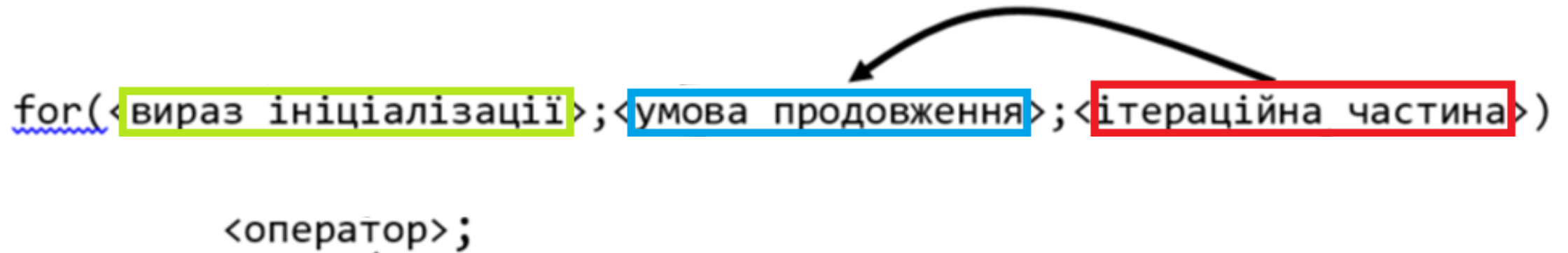
<оператор>;

num=3

```
for (let num = 1; num <= 15; num++) {  
  document.write('Hello')  
}
```

Схематичне зображення виконання оператора:

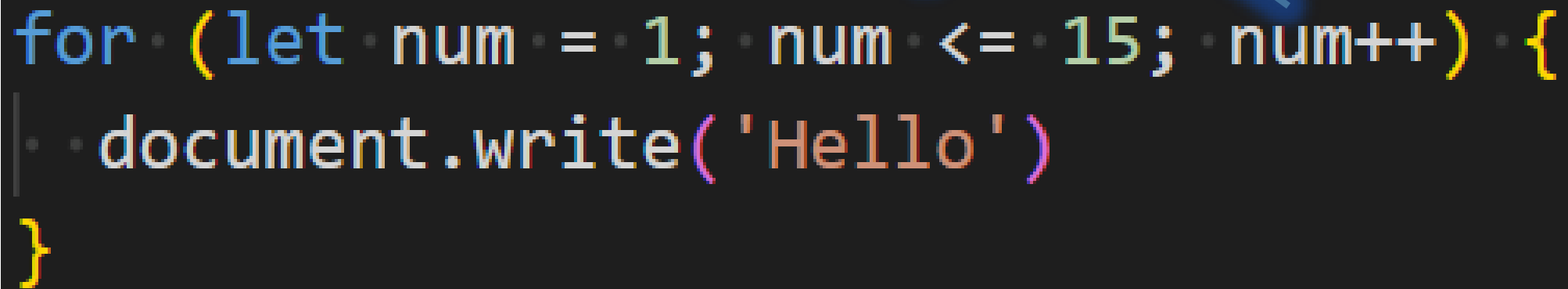
for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)
 <оператор>;



The diagram illustrates the structure of a for loop. The loop header is enclosed in parentheses and contains three parts: an initialization expression, a condition, and an iteration part. These parts are highlighted with colored boxes: green for the initialization expression, blue for the condition, and red for the iteration part. A curved arrow points from the iteration part back to the condition, indicating the loop's flow. Below the header is the body of the loop, which is a single statement followed by a semicolon.

num=3

```
for (let num = 1; num <= 15; num++) {  
    document.write('Hello')  
}
```



The diagram shows a specific example of a for loop in JavaScript. The loop header is highlighted with colored boxes: blue for the initialization expression 'let num = 1', green for the condition 'num <= 15', and yellow for the iteration part 'num++'. A curved arrow points from the iteration part back to the condition. A box labeled 'num=3' points to the condition, indicating the current state of the loop. The body of the loop is a single statement 'document.write('Hello')' followed by a closing brace.

Схематичне зображення виконання оператора:

for(<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)

<оператор>;

num=3

```
for (let num = 1; num <= 15; num++) {  
  document.write('Hello')  
}
```

Приклади випадків використання

Коли відома кількість повторень

Загальна форма	<pre>for (лічильник = 0; лічильник < кількість повторень ; лічильник ++) { оператор ; }</pre>
Приклад, розв'язаний з використанням for	<pre>//----- 100 разів вивести на екран слово «Мир» ---- for(let <u>i = 0</u> ; <u>i < 100</u> ; <u>i++</u>) { document.write('Мир'); }</pre>

Приклад. З клавіатури вводиться 5 пар цілих чисел. Якщо числа рівні, то вивести їх суму, інакше – добуток

Приклад. З клавіатури вводиться 5 пар цілих чисел. Якщо числа рівні, то вивести їх суму, інакше – добуток

5 разів повторити

{

 ввести перше число

 ввести друге число

 якщо числа рівні

 то вивести їх суму

 інакше

 вивести добуток

}

Приклад. З клавіатури вводиться 5 пар цілих чисел. Якщо числа рівні, то вивести їх суму, інакше – добуток

5 разів повторити
{

}

```
for (let i = 0; i < 5; i++) {
```

```
}
```

Приклад. З клавіатури вводиться 5 пар цілих чисел. Якщо числа рівні, то вивести їх суму, інакше – добуток

```
5 разів повторити
{
    ввести перше число
}
```

```
for (let i = 0; i < 5; i++) {  
    let num1=parseInt(prompt("Num1="))
```


Приклад. З клавіатури вводиться 5 пар цілих чисел. Якщо числа рівні, то вивести їх суму, інакше – добуток

```
5 разів повторити
{
    ввести перше число
    ввести друге число

}
```

```
for (let i = 0; i < 5; i++) {
    let num1=parseInt(prompt("Num1="))
    let num2=parseInt(prompt("Num2="))

}
```

Приклад. З клавіатури вводиться 5 пар цілих чисел. Якщо числа рівні, то вивести їх суму, інакше – добуток

5 разів повторити

```
{  
    ввести перше число  
    ввести друге число  
  
    якщо числа рівні  
        то вивести їх суму  
    інакше  
        вивести добуток  
}
```

```
for (let i = 0; i < 5; i++) {  
    let num1=parseInt(prompt("Num1="))  
    let num2=parseInt(prompt("Num2="))  
    if (num1==num2) {  
        document.write(`Sum=${num1+num2}`)  
    } else {  
        let product=num1*num2  
        document.write(`Prod=${product}`)  
    }  
}
```

Приклад. З клавіатури 6 разів генерується ціле число в межах від 1 до 10 і кожного разу дається можливість користувачу вгадати число.

Приклад. З клавіатури 6 разів генерується ціле число в межах від 1 до 10 і кожного разу дається можливість користувачу вгадати число.

Кількість_вгаданих=0

6 разів повторити

```
{  
    compNum = випадкове число від 1 до 10  
    userNum = питаємося від користувача число  
    якщо userNum дорівнює compNum  
        то Кількість_вгаданих++  
}
```

Виводимо кількість вгаданих

Приклад. З клавіатури 6 разів генерується ціле число в межах від 1 до 10 і кожного разу дається можливість користувачу вгадати число.

Кількість_вгаданих=0

```
<script>
    let guessedCount=0
```

</script>

Приклад. З клавіатури 6 разів генерується ціле число в межах від 1 до 10 і кожного разу дається можливість користувачу вгадати число.

Кількість_вгаданих=0

6 разів повторити

{

}

```
<script>
  let guessedCount=0

  }

</script>
```

Приклад. З клавіатури 6 разів генерується ціле число в межах від 1 до 10 і кожного разу дається можливість користувачу вгадати число.

Кількість_вгаданих=0

6 разів повторити
{

}

```
<script>
  let guessedCount=0
  for (let i = 0; i < 6; i++) {

  }

</script>
```

Приклад. З клавіатури 6 разів генерується ціле число в межах від 1 до 10 і кожного разу дається можливість користувачу вгадати число.

```
Кількість_вгаданих=0

6 разів повторити
{
    compNum = випадкове число від 1 до 10

}
```

```
<script>
    let guessedCount=0
    for (let i = 0; i < 6; i++) {
        let compNum=1+Math.floor(Math.random()*10)

    }

</script>
```


Приклад. З клавіатури 6 разів генерується ціле число в межах від 1 до 10 і кожного разу дається можливість користувачу вгадати число.

```
Кількість_вгаданих=0

6 разів повторити
{
  compNum = випадкове число від 1 до 10
  userNum = питаємося від користувача число

}
```

```
<script>
  let guessedCount=0
  for (let i = 0; i < 6; i++) {
    let compNum=1+Math.floor(Math.random()*10)
    let userNum=parseInt(prompt("Введіть число (від 1 до 10)"))

  }

</script>
```

Приклад. З клавіатури 6 разів генерується ціле число в межах від 1 до 10 і кожного разу дається можливість користувачу вгадати число.

```
Кількість_вгаданих=0

6 разів повторити
{
  compNum = випадкове число від 1 до 10
  userNum = питаємося від користувача число
  якщо userNum дорівнює compNum
    то Кількість_вгаданих++
}
```

```
<script>
  let guessedCount=0
  for (let i = 0; i < 6; i++) {
    let compNum=1+Math.floor(Math.random()*10)
    let userNum=parseInt(prompt("Введіть число (від 1 до 10)"))
    if (userNum==compNum) {
      guessedCount++
    }
    document.write(`User: ${userNum},   Comp:${compNum} <br>`)
  }

</script>
```

Приклад. З клавіатури 6 разів генерується ціле число в межах від 1 до 10 і кожного разу дається можливість користувачу вгадати число.

```
Кількість_вгаданих=0

6 разів повторити
{
    compNum = випадкове число від 1 до 10
    userNum = питаємося від користувача число
    якщо userNum дорівнює compNum
        то Кількість_вгаданих++
}

Виводимо кількість вгаданих
```

```
<script>
    let guessedCount=0
    for (let i = 0; i < 6; i++) {
        let compNum=1+Math.floor(Math.random()*10)
        let userNum=parseInt(prompt("Введіть число (від 1 до 10)"))
        if (userNum==compNum) {
            guessedCount++
        }
        document.write(`User: ${userNum},   Comp:${compNum} <br>`)
    }

    document.write(`Вгадано: ${guessedCount}`)
</script>
```

Поступово вводяться вартості 7 товарів. Знайти загальну вартість

Поступово вводяться вартості 7 товарів. Знайти загальну вартість

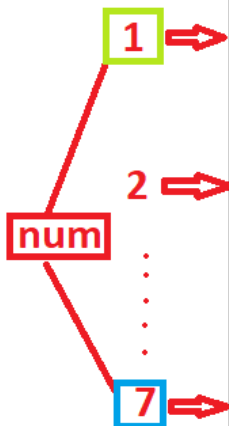
```
//Поступово вводяться вартості 7 товарів. Знайти загальну вартість
let price
let totalPrice = 0

1 ➡ price = parseFloat(prompt('Введіть вартість __ товару'))
totalPrice = totalPrice + price

2 ➡ price = parseFloat(prompt('Введіть вартість __ товару'))
totalPrice = totalPrice + price

//...

7 ➡ price = parseFloat(prompt('Введіть вартість __ товару'))
totalPrice = totalPrice + price
```



Поступово вводяться вартості 7 товарів із зазначенням номера товару. Знайти загальну вартість

```
//Поступово вводяться вартості 7 товарів. Знайти загальну вартість
let price
let totalPrice = 0

price = parseFloat(prompt('Введіть вартість 1 товару'))
totalPrice = totalPrice + price

price = parseFloat(prompt('Введіть вартість 2 товару'))
totalPrice = totalPrice + price

//...

price = parseFloat(prompt('Введіть вартість 7 товару'))
totalPrice = totalPrice + price
```

Поступово вводяться вартості 7 товарів із зазначенням номера товару. Знайти загальну вартість

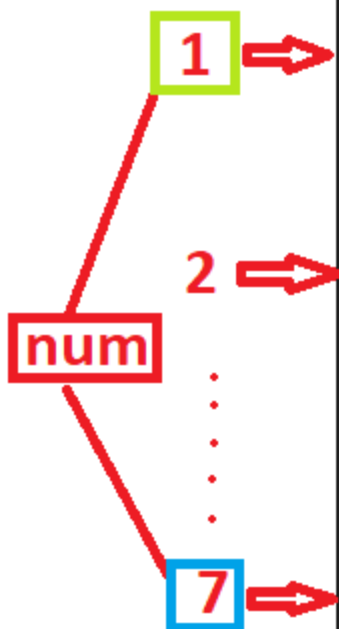
```
//Поступово вводяться вартості 7 товарів. Знайти загальну вартість
let price
let totalPrice = 0

price = parseFloat(prompt('Введіть вартість 1 товару'))
totalPrice = totalPrice + price

price = parseFloat(prompt('Введіть вартість 2 товару'))
totalPrice = totalPrice + price

//...

price = parseFloat(prompt('Введіть вартість 7 товару'))
totalPrice = totalPrice + price
```



Поступово вводяться вартості 7 товарів із зазначенням номера товару. Знайти загальну вартість

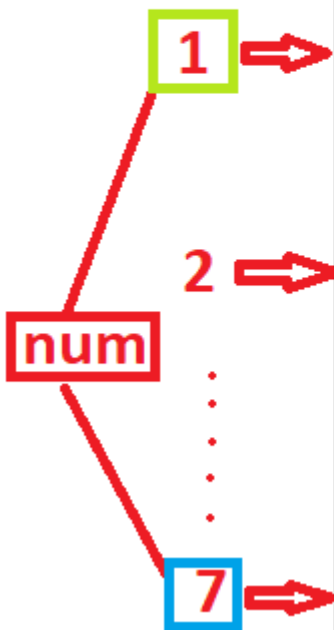
```
//Поступово вводяться вартості 7 товарів. Знайти загальну вартість
let price
let totalPrice = 0

price = parseFloat(prompt('Введіть вартість 1 товару'))
totalPrice = totalPrice + price

price = parseFloat(prompt('Введіть вартість 2 товару'))
totalPrice = totalPrice + price

//...

price = parseFloat(prompt('Введіть вартість 7 товару'))
totalPrice = totalPrice + price
```



Приклади випадків використання

Коли відомі початкове, кінцеве значення та крок зміни деякої величини (параметра)

Загальна форма	<pre>for (<u>параметр</u> = <u>поч.значення</u>; <u>параметр</u> <= <u>кін.значення</u>; <u>параметр</u> = <u>параметр</u> + <u>крок</u>) { оператор ; }</pre>
Приклад, розв'язаний з використанням for	<pre>//----- Вивести на екран усі числа кратні трьом <u>від 6 до 28</u>. for (var <u>i = 6</u>; <u>i <= 28</u>; <u>i=i+3</u>) { document.write(<u>i</u>); }</pre>

Задача. Вивести номери місяців другого півріччя (номери місяців від 7 до 12)

Місяць 7

Місяць 8

Місяць 9

Місяць 10

Місяць 11

Місяць 12

Задача. Вивести номери місяців другого півріччя (номери місяців від 7 до 12)

Місяць 7

Місяць 8

Місяць 9

Місяць 10

Місяць 11

Місяць 12

для кожного значення *номера* від 7 до 12 з кроком 1 {

}

Задача. Вивести номери місяців другого півріччя (номери місяців від 7 до 12)

Місяць 6

Місяць 7

Місяць 8

Місяць 9

Місяць 10

Місяць 11

Місяць 12

для кожного значення *номера* від 7 до 12 з кроком 1 {

}

```
for ( параметр = поч.значення; параметр <= кін.значення; параметр = параметр + крок )  
{  
    оператор ;  
}
```

Задача. Вивести номери місяців другого півріччя (номери місяців від 7 до 12)

Місяць 6

Місяць 7

Місяць 8

Місяць 9

Місяць 10

Місяць 11

Місяць 12

для кожного значення *номера* від 7 до 12 з кроком 1 {

}

```
for (let month = 7; month <= 12; month++) {  
  
}
```

```
for (параметр = поч.значення; параметр <= кін.значення; параметр = параметр + крок )  
{  
    оператор ;  
}
```

Задача. Вивести номери місяців другого півріччя (номери місяців від 6 до 12)

Місяць 6 Місяць 7 Місяць 8 Місяць 9 Місяць 10 Місяць 11 Місяць 12

```
<style>
  div {
    display: inline-block;
    border: 2px solid red;
    margin: 10px;
  }
</style>
```

для кожного значення **номера** від 7 до 12 з кроком 1 {
вивести **номер**

}

```
for (let month = 7; month <= 12; month++) {
  document.write(`<div>Місяць ${month}</div>`)
}
```

```
for (параметр = поч.значення; параметр <= кін.значення; параметр = параметр + крок)
{
  оператор ;
}
```

Задача. Вивести суми сплаченого кредиту. Початковий внесок 5000грн, кінцевий 15000 грн, а щомісячний внесок 2000грн.

5000грн 7000грн 9000грн 11000грн 13000грн 15000грн

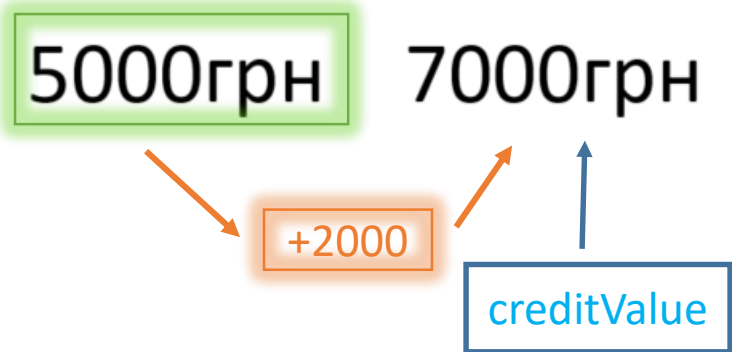
Задача. Вивести суми сплаченого кредиту. Початковий внесок 5000грн, кінцевий 15000 грн, а щомісячний внесок 2000грн.

5000грн

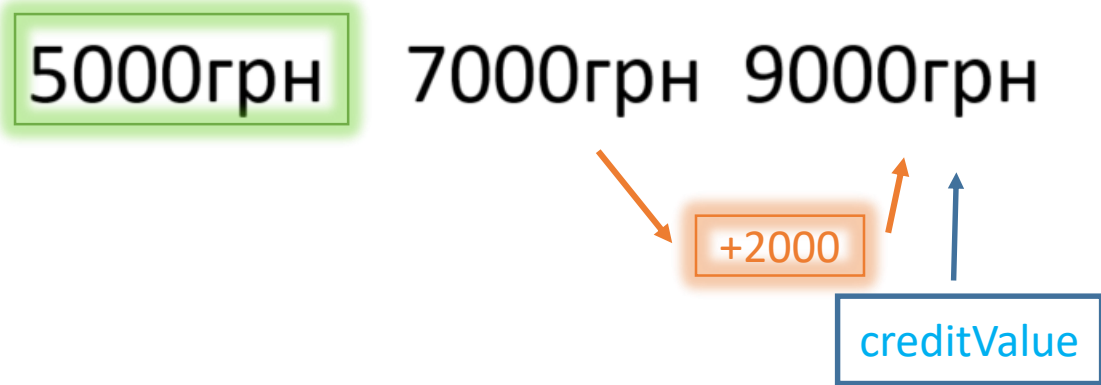


creditValue

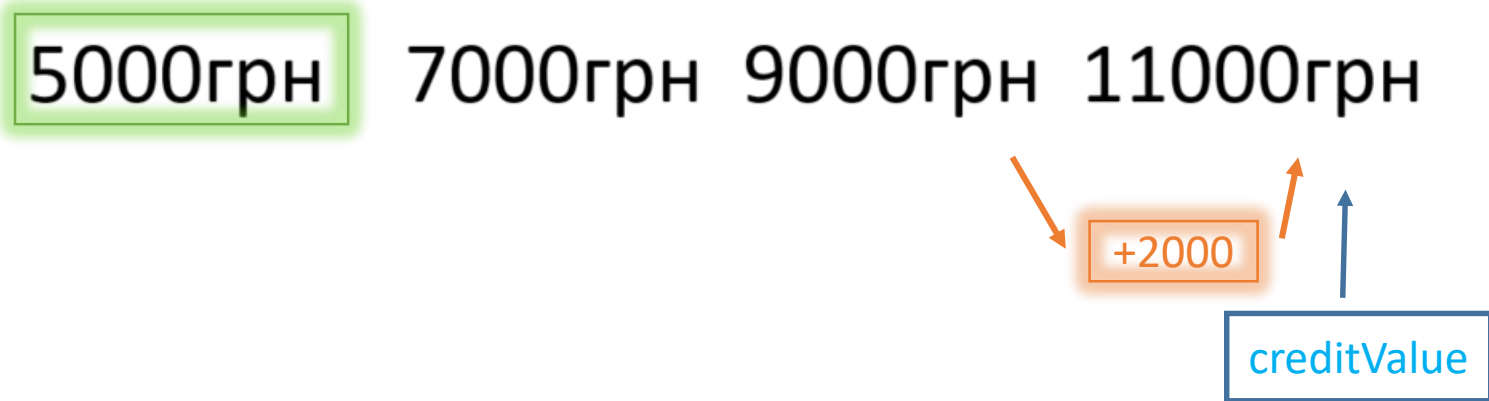
Задача. Вивести суми сплаченого кредиту. Початковий внесок 5000грн, кінцевий 15000 грн, а щомісячний внесок 2000грн.



Задача. Вивести суми сплаченого кредиту. Початковий внесок 5000грн, кінцевий 15000 грн, а щомісячний внесок 2000грн.

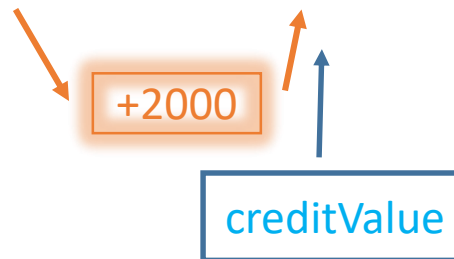


Задача. Вивести суми сплаченого кредиту. Початковий внесок 5000грн, кінцевий 15000 грн, а щомісячний внесок 2000грн.

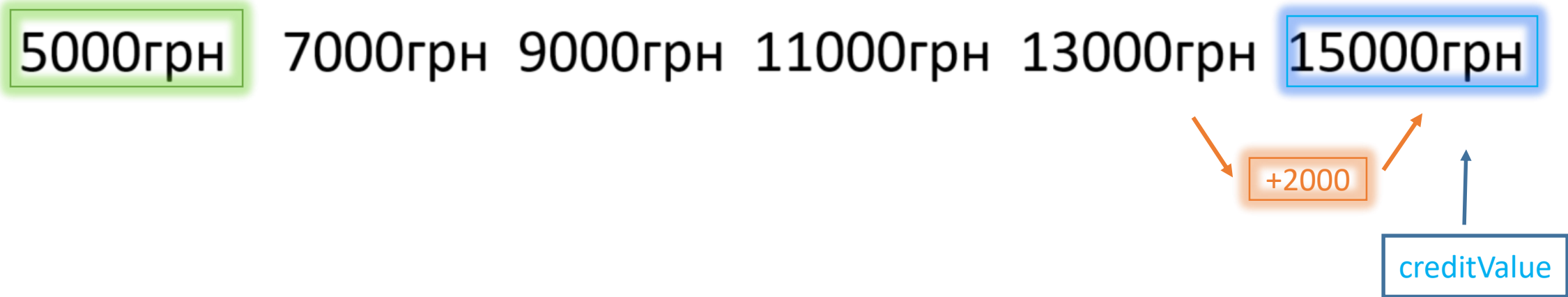


Задача. Вивести суми сплаченого кредиту. Початковий внесок 5000грн, кінцевий 15000 грн, а щомісячний внесок 2000грн.

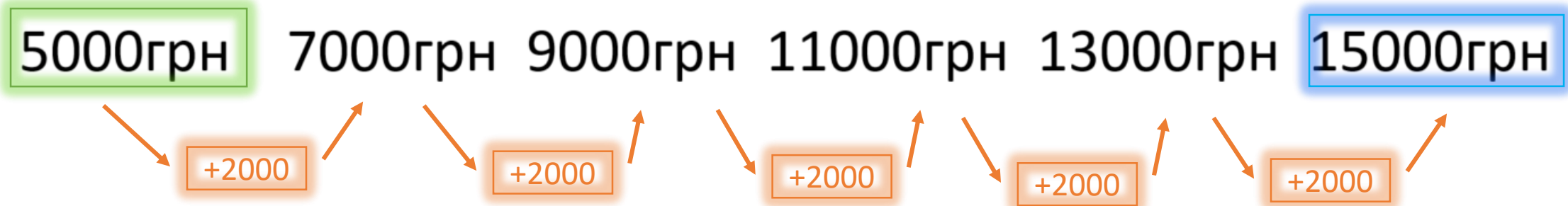
5000грн 7000грн 9000грн 11000грн 13000грн



Задача. Вивести суми сплаченого кредиту. Початковий внесок 5000грн, кінцевий 15000 грн, а щомісячний внесок 2000грн.



Задача. Вивести суми сплаченого кредиту. Початковий внесок 5000грн, кінцевий 15000 грн, а щомісячний внесок 2000грн.



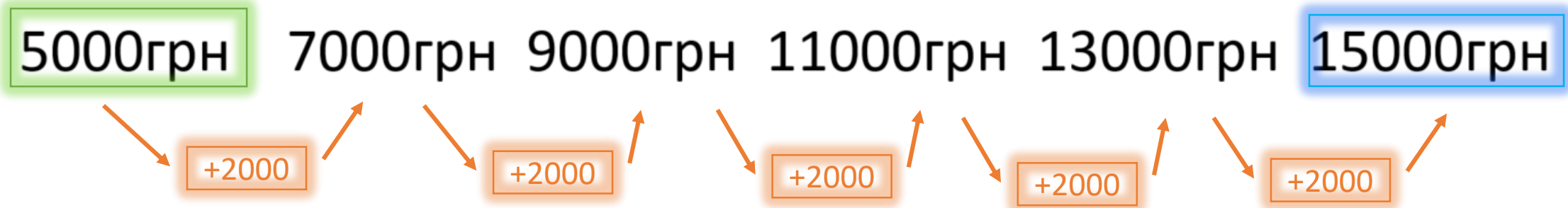
для кожного значення
creditValue від 5000 до 15000 з кроком 2000 {

}
}



```
for (параметр = поч.значення; параметр <= кін.значення; параметр = параметр + крок )  
{  
    оператор ;  
}
```

Задача. Вивести суми сплаченого кредиту. Початковий внесок 5000грн, кінцевий 15000 грн, а щомісячний внесок 2000грн.



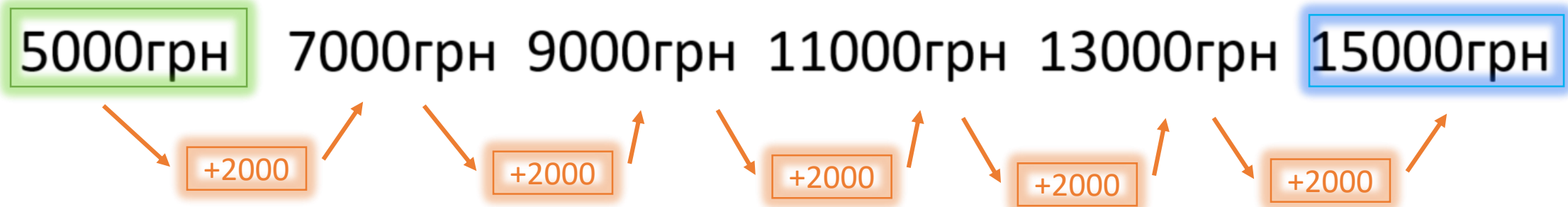
для кожного значення
creditValue від 5000 до 15000 з кроком 2000 {

}
}

```
for (let creditValue = 5000; creditValue <= 15000; creditValue += 2000) {  
  
}  
}
```

```
for (параметр = поч.значення; параметр <= кін.значення; параметр = параметр + крок )  
{  
    оператор ;  
}
```

Задача. Вивести суми сплаченого кредиту. Початковий внесок 5000грн, кінцевий 15000 грн, а щомісячний внесок 2000грн.



для кожного значення
creditValue від 5000 до 15000 з кроком 2000 {
вивести *creditValue*
}

```
for (let creditValue = 5000; creditValue <= 15000; creditValue += 2000) {  
    document.write(`<div>${creditValue}грн </div>`)  
}
```

```
for (параметр = поч.значення; параметр <= кін.значення; параметр = параметр + крок )  
{  
    оператор ;  
}
```


Приклад. «Тренажер додавання». Розробити програму для перевірки знань з додавання цілих чисел в межах від 1 до 3(перебрати усі можливі комбінації додавання цілих чисел (тобто перше число перебрати від 1 до 3 і для кожного першого числа перебрати числа від 1 до 5). Іншими словами:

1+1, 1+2, 1+3

2+1, 2+2, 2+3

3+1, 3+2, 3+3

Приклад. «Тренажер додавання». Розробити програму для перевірки знань з додавання цілих чисел в межах від 1 до 3(перебрати усі можливі комбінації додавання цілих чисел (тобто перше число перебрати від 1 до 3 і для кожного першого числа перебрати числа від 1 до 3). Іншими словами:

1+1, 1+2, 1+3

2+1, 2+2, 2+3

3+1, 3+2, 3+3

```
перше число змінюється від 1 до 3 // треба повторювати
{
    друге число змінюється від 1 до 3 //треба повторювати
    {
        виводимо повідомлення «Чому дорівнює перше + друге »
        вводимь відповідь користувача
        якщо відповідь дорівнює сумі чисел (відповідь правильна)
            то кажемо «Ок»
        інакше
            кажемо «помилка»
    }
}
```

Приклад. «Тренажер додавання». Розробити програму для перевірки знань з додавання цілих чисел в межах від 1 до 3(перебрати усі можливі комбінації додавання цілих чисел (тобто перше число перебрати від 1 до 3 і для кожного першого числа перебрати числа від 1 до 3). Іншими словами:

1+1, 1+2, 1+3

2+1, 2+2, 2+3

3+1, 3+2, 3+3

```
перше число змінюється від 1 до 3 // треба повторювати
{

}

}
```



Приклад. «Тренажер додавання». Розробити програму для перевірки знань з додавання цілих чисел в межах від 1 до 3(перебрати усі можливі комбінації додавання цілих чисел (тобто перше число перебрати від 1 до 3 і для кожного першого числа перебрати числа від 1 до 3). Іншими словами:

1+1, 1+2, 1+3

2+1, 2+2, 2+3

3+1, 3+2, 3+3

перше число змінюється від 1 до 3 // треба повторювати

{

}

```
for (let num1 = 1; num1 <=3; num1++) {
```

```
}
```

Приклад. «Тренажер додавання». Розробити програму для перевірки знань з додавання цілих чисел в межах від 1 до 3(перебрати усі можливі комбінації додавання цілих чисел (тобто перше число перебрати від 1 до 3 і для кожного першого числа перебрати числа від 1 до 3). Іншими словами:

1+1, 1+2, 1+3

2+1, 2+2, 2+3

3+1, 3+2, 3+3

```
перше число змінюється від 1 до 3 // треба повторювати
{
    друге число змінюється від 1 до 3 //треба повторювати
    {

    }
}
```

```
for (let num1 = 1; num1 <=3; num1++) {
```

```
}
```

Приклад. «Тренажер додавання». Розробити програму для перевірки знань з додавання цілих чисел в межах від 1 до 3(перебрати усі можливі комбінації додавання цілих чисел (тобто перше число перебрати від 1 до 3 і для кожного першого числа перебрати числа від 1 до 3). Іншими словами:

1+1, 1+2, 1+3

2+1, 2+2, 2+3

3+1, 3+2, 3+3

```
перше число змінюється від 1 до 3 // треба повторювати
{
    друге число змінюється від 1 до 3 //треба повторювати
    {

    }
}
```

```
for (let num1 = 1; num1 <=3; num1++) {
    for (let num2 = 1; num2 <=3; num2++)
    {
    }
}
```

Приклад. «Тренажер додавання». Розробити програму для перевірки знань з додавання цілих чисел в межах від 1 до 3(перебрати усі можливі комбінації додавання цілих чисел (тобто перше число перебрати від 1 до 3 і для кожного першого числа перебрати числа від 1 до 3). Іншими словами:

1+1, 1+2, 1+3

2+1, 2+2, 2+3

3+1, 3+2, 3+3

```
перше число змінюється від 1 до 3 // треба повторювати
{
    друге число змінюється від 1 до 3 //треба повторювати
    {
        виводимо повідомлення «Чому дорівнює перше + друге »
        вводимо відповідь користувача

    }
}
```

```
for (let num1 = 1; num1 <=3; num1++) {
    for (let num2 = 1; num2 <=3; num2++)
    {
        let userAns=parseInt(prompt(`${num1}+${num2}=`))

    }
}
```

Приклад. «Тренажер додавання». Розробити програму для перевірки знань з додавання цілих чисел в межах від 1 до 3(перебрати усі можливі комбінації додавання цілих чисел (тобто перше число перебрати від 1 до 3 і для кожного першого числа перебрати числа від 1 до 3). Іншими словами:

1+1, 1+2, 1+3

2+1, 2+2, 2+3

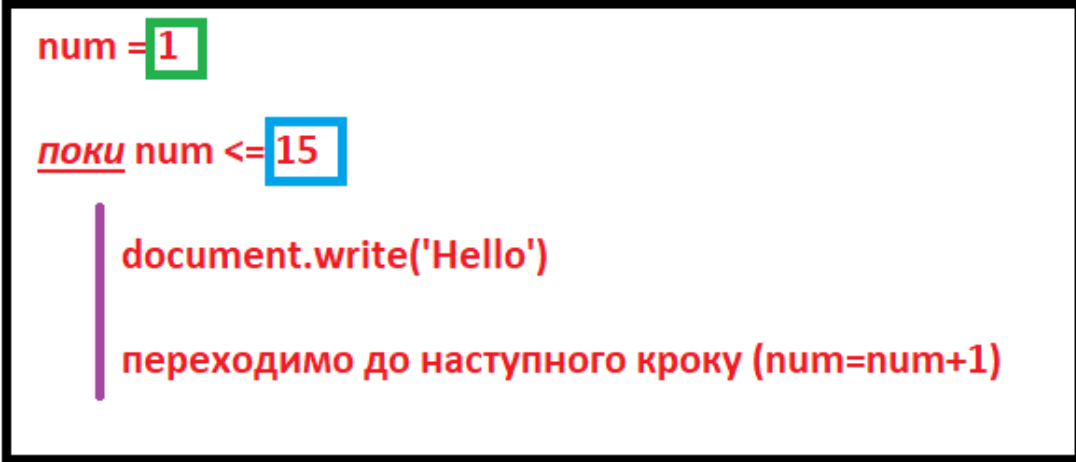
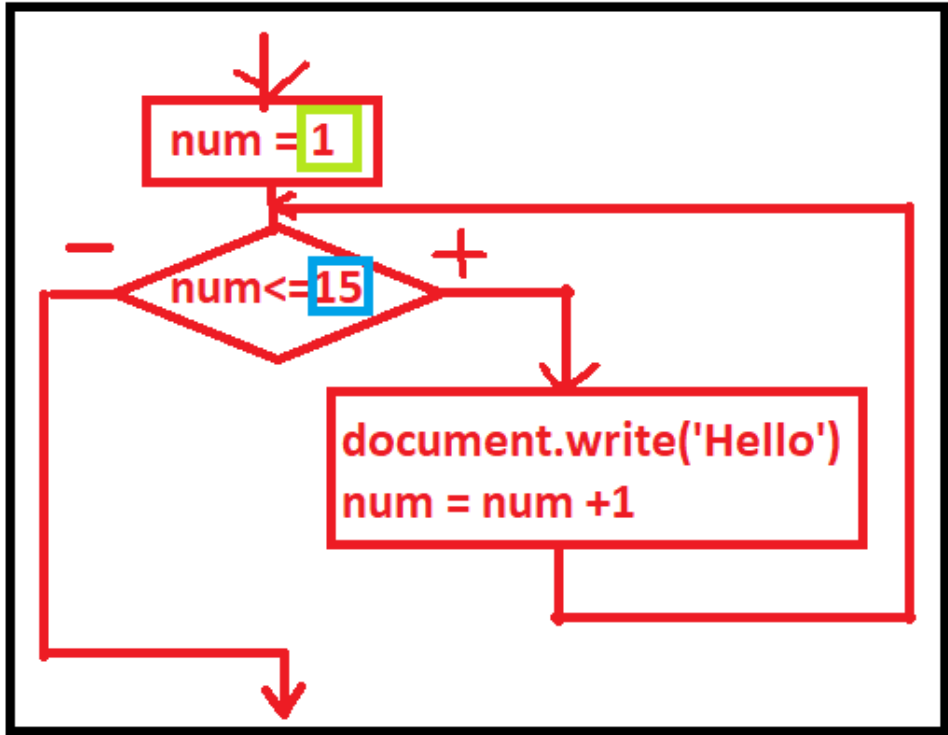
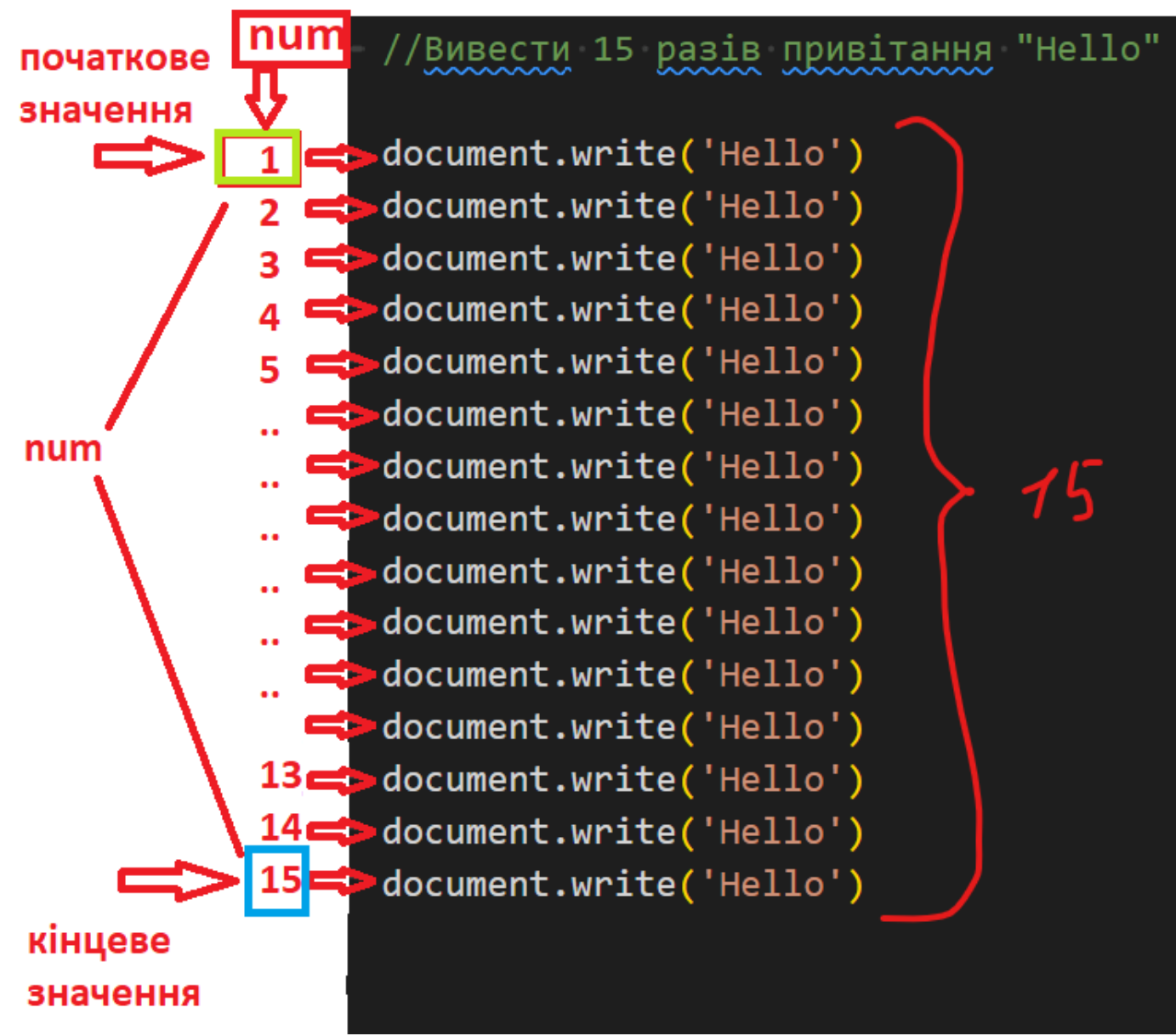
3+1, 3+2, 3+3

```
перше число змінюється від 1 до 3 // треба повторювати
{
    друге число змінюється від 1 до 3 //треба повторювати
    {
        виводимо повідомлення «Чому дорівнює перше + друге »
        вводимо відповідь користувача
        якщо відповідь дорівнює сумі чисел (відповідь правильна)
            то кажемо «Ок»
        інакше
            кажемо «помилка»
    }
}
```

```
for (let num1 = 1; num1 <=3; num1++) {
    for (let num2 = 1; num2 <=3; num2++)
    {
        let userAns=parseInt(prompt(`${num1}+${num2}=`))
        if (userAns==num1+num2) {
            alert('ok')
        }
        else
        {
            alert('Error')
        }
    }
}
```

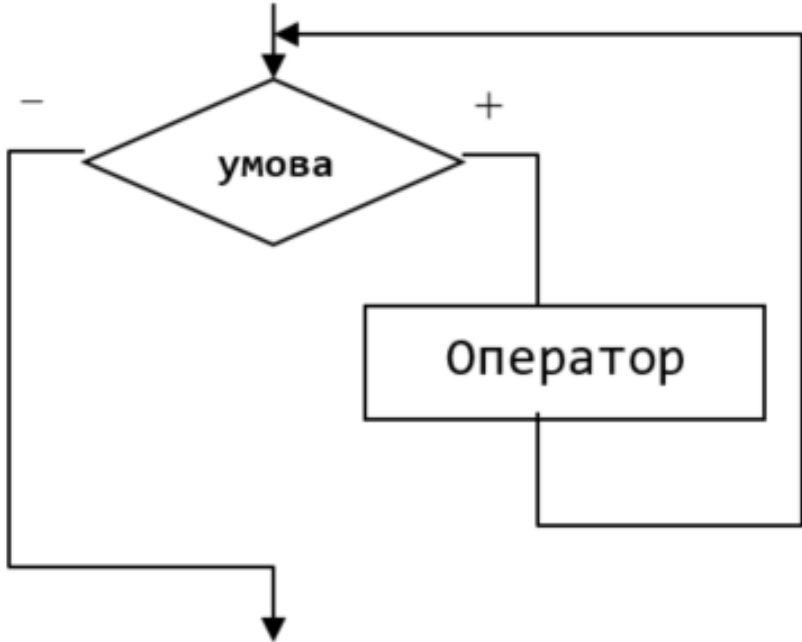

Приклад*. «Тренажер множення». Розробити програму, яка випадковим чином (4 рази) генерує перше число в межах від 1 до 9, друге число від 1 до 9 і перевірити чи знає користувач, чому дорівнює результат їх множення. Підрахувати кількість правильних відповідей.

Для чого потрібні цикли?



Оператор циклу з передумовою while

Оператор while циклічно виконує оператор (своє тіло) до тих пір, поки умова виконується (логічний вираз приймає значення true).

Програмна структура	Аналог на мові блок-схем	Приклад.
<pre>while(умова) оператор ;</pre>		<p>Знаходити суму чисел, які вводить користувач, поки сума є меншою за 100.</p> <pre>var sum=0; while(sum<100) sum=sum+ +prompt('number=','0');</pre>

поки виконується
ця умова

Повторювати
ці команди

Програмна структура	Аналог на мові блок-схем
<pre>while(умова) оператор ;</pre>	<pre>graph TD; Entry(()) --> Uмова{умова}; Uмова -- "-" --> Exit(()); Uмова -- "+" --> Оператор[Оператор]; Оператор --> Uмова;</pre>

Вводимо початкову кількість булочок, які треба продати. Поки залишаються булочки продавати їх клієнтам

Повторювати поки кількість булочок > 0

(

питаємось скільки користувач хоче купити булочок

якщо є така кількість

то продаємо

інакше повідомляємо, що такої кількості немає

)

Вводимо початкову кількість булочок, які треба продати. Поки залишаються булочки продавати їх клієнтам

Повторювати поки кількість булочок > 0

(

питаємось скільки користувач хоче купити булочок

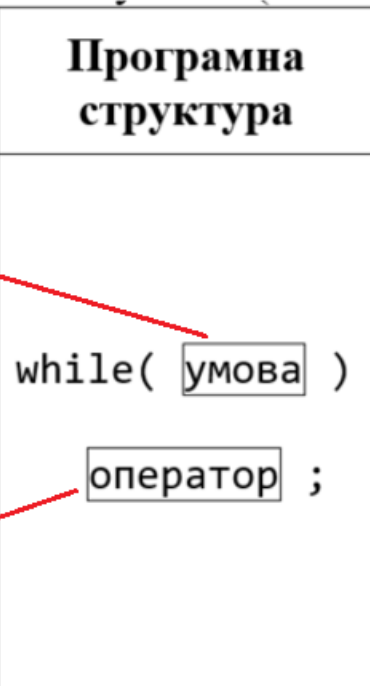
якщо є така кількість

то продаємо

інакше повідомляємо, що такої кількості немає

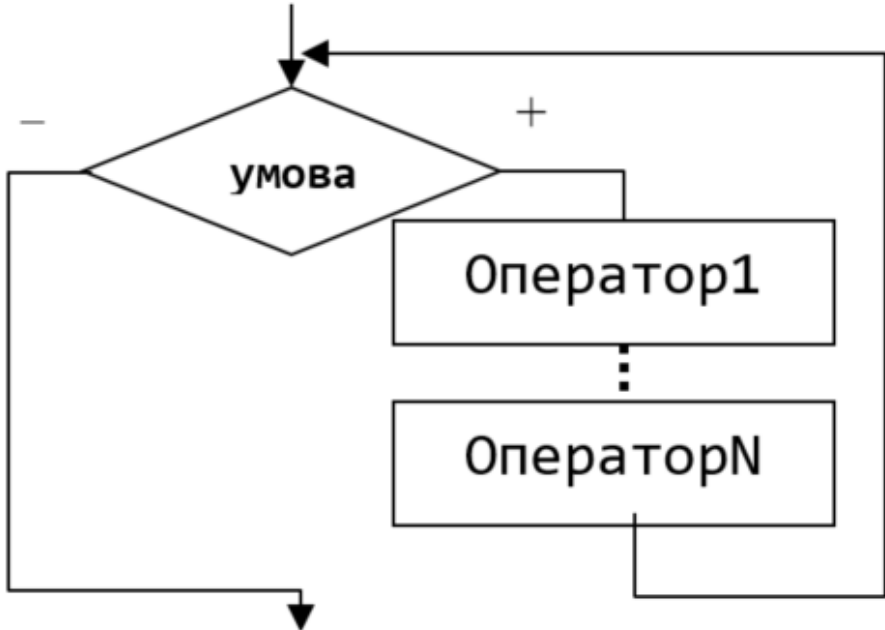
)

поки виконується
ця умова



Повторювати
ці команди

Якщо тіло циклу складається з більше ніж одного оператора, то необхідно використати складений оператор (записати ці оператори у фігурних дужках).

Програмна структура	Аналог на мові блок-схем	Приклад. Знаходити суму і добуток чисел, які вводить користувач, поки сума є меншою за 100
<pre>while(умова) { оператор1 ; операторN ; }</pre>		<pre>let sum=0; let mult=1; while(sum<=100) { let number = +prompt('number=', '0'); sum = sum + number; mult = mult * number; }</pre>

Приклад*. З клавіатури вводяться два числа N і M ($N < M$). Вивести а екран числа

$N \dots M$

$N+1 \dots M-1$

$N+2 \dots M-2$

$N+3 \dots M-3$

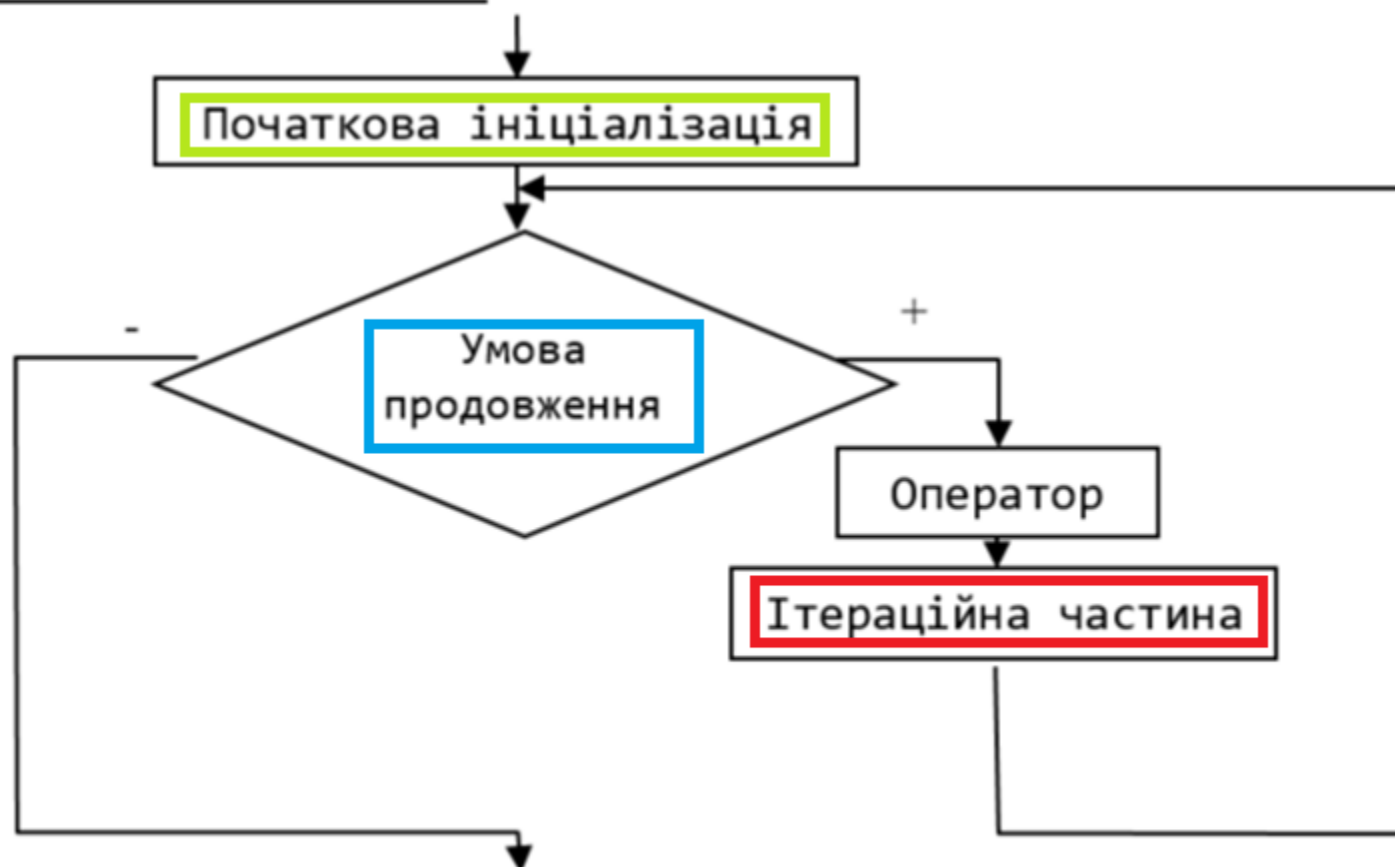
Цикл з параметром for

Найчастіше у випадку, коли повторення потрібно робити у залежності від зміни деякого параметра або ж у випадку, коли потрібно повторювати тіло циклу деяку кількість разів використовують цикл з параметром `for`.

Загальний вигляд:

```
for (<вираз ініціалізації>; <умова продовження>; <ітераційна частина>)  
    <оператор>;
```

Аналог на мові блок-схем:



```
for ( параметр = поч.значення; параметр <= кін.значення; параметр = параметр + крок )  
{  
    оператор ;  
}
```

```
let num = 1  
while (num <= 7) {  
    price = parseFloat(prompt(`Введіть вартість ${num} товару`))  
    totalPrice = totalPrice + price  
    num++  
}
```

```
for (let num = 1; num <= 7; num++) {  
    price = parseFloat(prompt(`Введіть вартість ${num} товару`))  
    totalPrice = totalPrice + price  
}
```

num можна використати і після циклу

num можна використати тільки всередині циклу !!!!!!!

Поступово вводяться вартості 7 товарів із зазначенням номера товару. Знайти загальну вартість

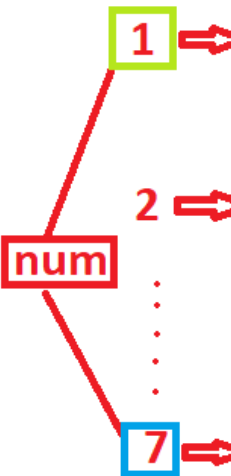
```
//Поступово вводяться вартості 7 товарів. Знайти загальну вартість
let price
let totalPrice = 0

price = parseFloat(prompt('Введіть вартість 1 товару'))
totalPrice = totalPrice + price

price = parseFloat(prompt('Введіть вартість 2 товару'))
totalPrice = totalPrice + price

//...

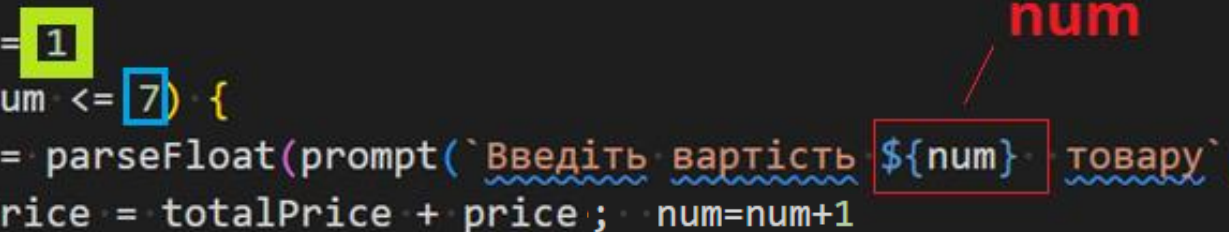
price = parseFloat(prompt('Введіть вартість 7 товару'))
totalPrice = totalPrice + price
```



```
//Поступово вводяться вартості 7 товарів. Знайти загальну вартість
let price
let totalPrice = 0

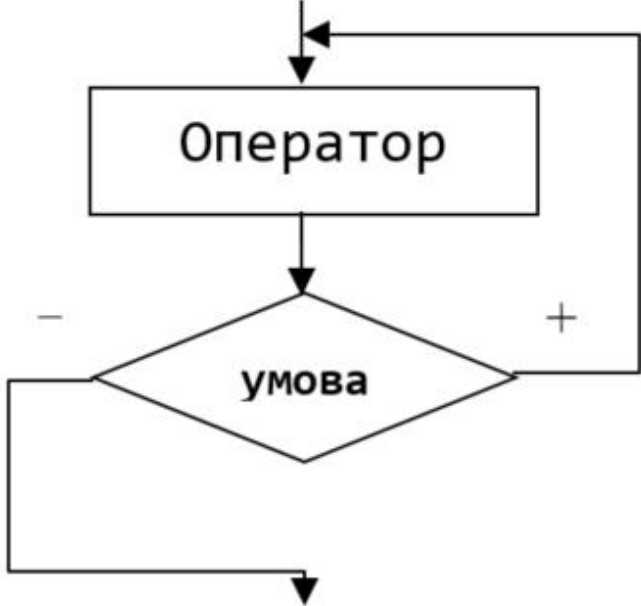
let num = 1
while (num <= 7) {
  price = parseFloat(prompt(`Введіть вартість ${num} товару`))
  totalPrice = totalPrice + price; num=num+1
}

document.write(`Загальна вартість ${totalPrice}`)
```



Цикл з післяумовою do-while

Оператор циклу do-while відрізняється від оператора while тим, що перевірка умови виконується не до, а після виконання тіла циклу (оператора). А тому у операторі циклу do-while тіло циклу виконається принаймні один раз. Як і в циклі з передумовою, тіло виконується поки умова вірна.

Програмна структура	Аналог на мові блок-схем	Приклад. Знаходити суму чисел, які вводить користувач поки не буде введено 0. Зауважимо, що принаймні одне число треба ввести (принаймні один раз тіло циклу потрібно виконати).
<pre>do { оператор ; } while (умова);</pre>		<pre>var sum=0; var number; do { number= +prompt ("number=", "0") ; sum=sum+number; } while(number!=0);</pre>

Приклад*. Оленка має 0 грн. і хоче назбирати на телефон (10000 грн.). Для цього кожного разу допомагає мамі і отримує гонорар - випадкова величина від 1 до 200 грн. Підрахувати скільки разів треба допомагати Оленці поки не назбирає на телефон.

Приклад*. Оленка має М грн. (вводиться з клавіатури) і хоче назбирати на телефон (10000 грн.). Для цього кожного разу допомагає мамі і отримує гонорар - випадкова величина від 1 до 200 грн. Підрахувати скільки разів треба допомагати Оленці поки не назбирає на телефон.

З клавіатури вводяться поступово двоцифрові числа. Знаходимо їх добуток поки не буде введено 0.


З клавіатури вводяться числа до тих пір, поки не буде введено 3 парних.

Оператор `break`

Вийти з циклу можна не тільки при перевірці умови але й, взагалі, в будь-який момент. Цю можливість забезпечує оператор **break**.

Приклад. Поступово вводити 10 цілих чисел і знайти їх суму. Якщо буде введено некоректне число, то припинити знаходження суми.

```
let sum = 0
for (let i = 0; i < 10; i++) {
  let num = parseInt(prompt('Number = '))
  if (!isFinite(num)) // Якщо значення некоректне - припинити
    break
  sum += num
}
document.write(`Sum = ${sum}`)
```



Приклади*. З рази дати можливість ввести пароль (правильний пароль «123»).

break з міткою

Буває потрібно вийти одночасно з декількох рівнів циклу. Звичний виклик `break` не може перервати два цикли відразу. Для цього існують мітки.

Мітка має вигляд унікального ідентифікатора (як назва змінної), після якого ставлять двокрапку

Приклад. Користувач кожного дня протягом 7 тижнів купляв товар за ціною, що могла змінюватись щодня. Знайти скільки всього заплатив користувач. Якщо користувач припиняє введення (натиснув на «Відміна» під час введення), то вивести поточну суму і припинити сумування.

exit : - мітка

цикл, який буде перервано
з використанням мітки **exit**

```
let sum = 0
exit: for (let i = 0; i < 3; i++) {
  for (let j = 0; j < 7; j++) {
    let price = prompt(`Тиждень ${i}
    день ${j}`)
    console.log(price)
    if (price === null) break exit

    sum += parseFloat(price)
  }
}

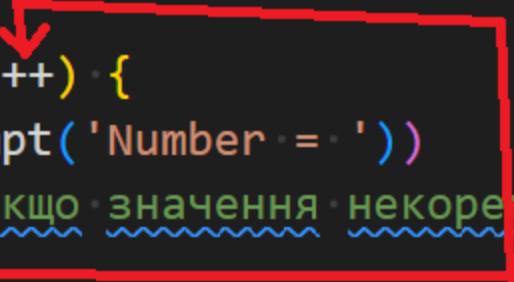
document.write(`Sum = ${sum}`)
```

Оператор continue

Директива **continue** припиняє виконання поточної ітерації циклу та виконує перехід до наступної. Вона також *може застосовуватись з мітками*.

Приклад. Поступово вводити 10 цілих чисел і знайти їх суму. Якщо буде введено некоректне число, то ігнорувати його і перейти до введення наступного числа (в результаті кількість чисел буде меншою). Вивести знайдену суму.

```
let sum = 0
for (let i = 0; i < 10; i++) {
  let num = parseInt(prompt('Number = '))
  if (!isFinite(num)) //якщо значення некоректне - перейти до введення наступного числа
    continue
  sum += num
}
document.write(`Sum = ${sum}`)
```



Приклад. Комп'ютер поступово генерує 3 випадкові числа (від 1 до 5). Для вгадування кожного числа користувач має 4 спроби. Підрахувати кількість вгаданих чисел.

```
const MIN_NUM = 1
const MAX_NUM = 5
let guessedNumber = 0
mainLoop: for (let i = 1; i <= 3; i++) {
  const compNum = Math.floor(
    MIN_NUM + Math.random() * (MAX_NUM - MIN_NUM + 1)
  )
  for (let j = 0; j < 4; j++) {
    let userNum = parseInt(
      prompt(`Вгадування ${i}-го числа. Ваша версія: `)
    )
    if (userNum === compNum) {
      alert('Вгадали')
      guessedNumber++
      continue mainLoop
    } else alert('Не вгадали')
  }
}
document.write(`Вгадано чисел : ${guessedNumber}`)
```