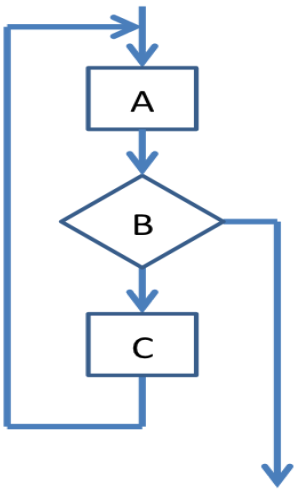
Estruturas de Repetição

Loop

Já vimos que programação é, basicamente, sobre dar instruções para o computador resolver um problema. Mas será que precisamos escrever todas as instruções, mesmo quando é um processo repetitivo?

Imagine que precisamos imprimir números de 1 a 10. Poderíamos fazer isso escrevendo 10 vezes o comando console.log, porém isso não parece muito prático, não é?

```
console.log(1)
console.log(2)
console.log(3)
console.log(4)
console.log(5)
console.log(6)
console.log(7)
console.log(8)
console.log(9)
console.log(10)
```



Para resolver esse problema em programação, temos o loop. Também conhecido como ciclo ou laço de repetição, ele executa um trecho de código quantas vezes forem necessárias!

No fluxograma ao lado, a instrução A é executada, e a instrução B é uma condicional que pergunta se o loop deve continuar se repetindo. Caso a condição seja verdadeira, a instrução C será executada e voltará ao início do procedimento.

Toda estrutura de repetição terá sempre:

- um ponto de partida (inicialização);
- uma condição que controla as repetições;
- uma instrução que deve ser repetida;
- um incremento para a condição.

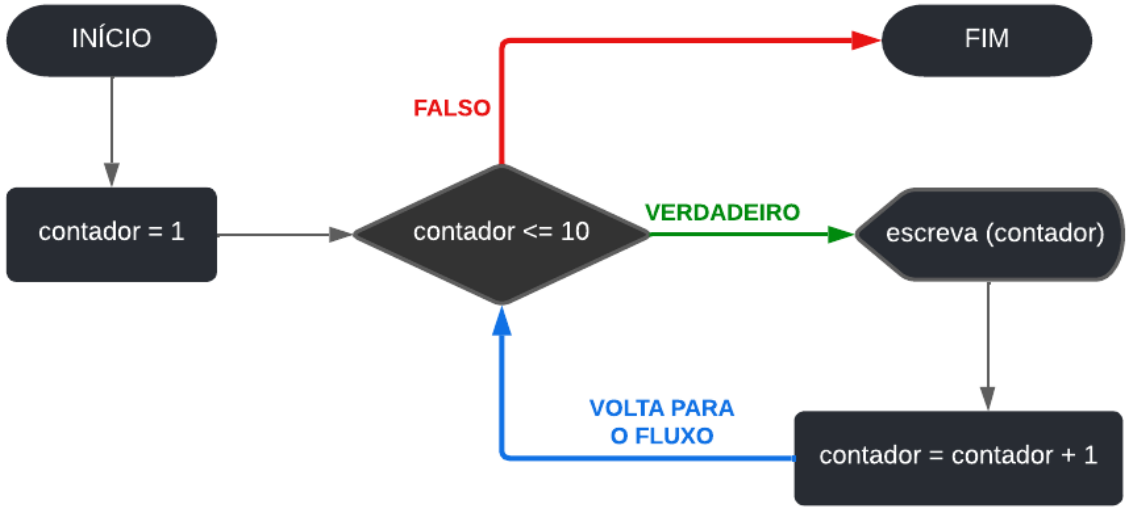
No ponto de partida, definimos uma variável de controle que começa com um valor inicial e que serve de **contador** para a quantidade de vezes que queremos executar o código.

A condição que controla as repetições, verifica se o código será ou não executado, através de uma expressão relacional que retorna os valores "**verdadeiro**" ou "**falso**". Quando a expressão retorna "verdadeiro", o loop é executado. Quando o valor retornado é "falso", o loop se encerra.

A instrução que deve ser repetida, nada mais é que o **trecho de código** que precisamos executar um determinado número de vezes. Esse trecho de código pode ser composto por uma ou várias linhas de código, podendo inclusive conter outras estruturas condicionais ou de repetição.

E por fim, o **incremento** é uma forma de atualizarmos o valor do contador a cada ciclo do loop, desta forma, garantimos que em algum momento o valor do contador não atenderá mais a condição de repetição e assim encerrará o loop.

Vejamos um exemplo desse código em fluxograma:



No fluxograma acima identificamos que o **ponto de partida** acontece quando definimos **“contador = 1”**. Já a condição que **controla o fluxo** de repetição está definida na expressão **“contador <= 10”**, que enquanto retornar verdadeiro entra no loop para executar o **comando** definido como **“escreva (contador)”**.

Observe que após a execução do comando, o contador recebe um **“incremento”** que aumenta seu valor em +1 para depois retornar para o fluxo. Quando a condição não for mais aceita, o fluxo será desviado para o fim do algoritmo.

Estruturas de Repetição

Loop For

O Javascript tem diversos comandos que podemos utilizar para trabalhar com loops. Todavia, vamos aprender alguns deles nesse primeiro momento e ao longo da sua trajetória na nossa formação, veremos as demais possibilidades.

Iniciaremos apresentando a instrução **“for()”**, que na sua tradução literal, significa “para” e traz a ideia de que “para cada ocorrência que satisfaça uma condição, uma tarefa precisa ser realizada”. Essa é a estrutura de repetição mais comumente utilizada por programadores toda vez que precisamos repetir uma ação várias vezes.

Sintaxe do for:

Para que o **for()** funcione corretamente, precisamos definir dentro dos seus parênteses, os 3 parâmetros, separados por ponto e vírgula “**;**”, que correspondem às informações sobre a **inicialização**, a **condição** e o **incremento**.

Além disso, os **comandos** a serem executados precisam estar dentro do bloco do for(), entre o par de chaves “**{ }**”, como mostra o exemplo a seguir:

```
for (inicialização; condição; incremento) {
     comando A
     comando B
     ...
}
```

Vejamos cada uma dessas partes:

inicialização: declaramos e atribuímos um valor inicial a uma variável que servirá de base para controlar o fluxo de repetição, também chamado de contador. O mais legal é que podemos decidir em qual número o contador deve iniciar.
Exemplo: inteiro contador = 1

condição: definimos a regra para que o loop continue acontecendo, ou seja, para cada vez que o programa avaliar a condição estabelecida e ela for aceita (retornar true), o programa entra no bloco e realiza a ação mais uma vez. O ciclo só será interrompido quando a condição não for mais aceita.
Exemplo: contador < 10

incremento: para cada vez que o loop é executado, nós alteramos o número do contador, adicionando +1 ao seu valor.
Exemplo: contador = contador + 1.

Em Portugol, esse código pode ser escrito da seguinte forma:

O código acima deverá imprimir os números de 1 até 10 como resultado. Todavia, ainda podemos melhorar esse código escrevendo o incremento de uma forma mais curta, substituindo "**contador = contador + 1**" por "**contador++**". Esse mesmo trecho de código escrito em Javascript, ficaria assim:

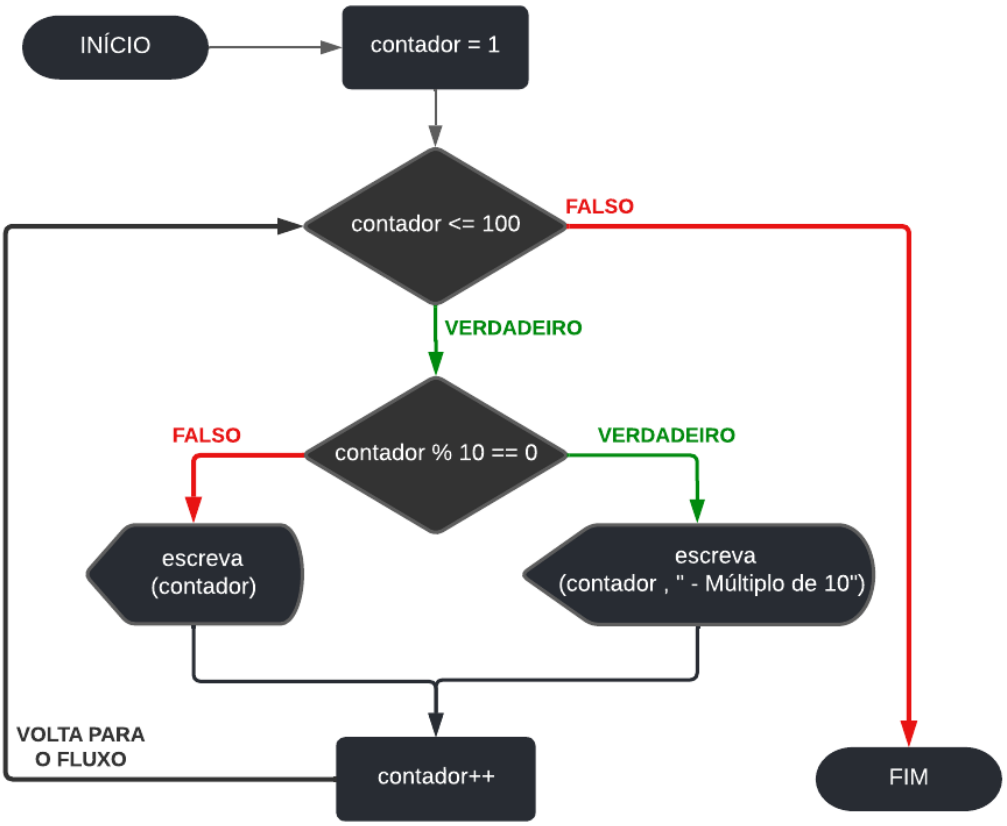
```
for (var contador = 1; contador <= 10; contador++) {
    // código que deve se repetir
    console.log(contador)
}
```

Dessa forma, o contador inicia com valor 1; para cada repetição ele deve adicionar +1 (incremento); e continua repetindo enquanto a condição (contador <= 10) for verdadeira.

Vamos aplicar esse conhecimento em outro exemplo:

Situação Problema 1:

Uma Professora de Matemática está com dificuldades para ensinar os múltiplos de 10 para sua turma, então ela decidiu utilizar a programação para atrair a atenção dos seus alunos. Porém, ela não sabe programar e pediu nossa ajuda para criar uma aplicação que imprima os valores de 1 a 100, e para cada vez que a impressão encontrar um valor múltiplo de 10, imprima também, a frase: " - Múltiplo de 10".



Observe que iniciamos nosso fluxo definindo o valor 1 para o contador. Em seguida, entramos no loop que verifica se o contador é menor ou igual a 100; como a resposta para esta condição é verdadeira, entramos na condição que testa se o contador é múltiplo de 10 através da expressão "**contador % 10 == 0**".

Nesse caso, o operador "**%**" vai dividir o contador por 10 e retornar o resto da divisão, e caso o resultado seja zero (o que significa que o contador nesse momento é múltiplo de 10) deve imprimir a frase correspondente e aumentar o contador em +1.

O fluxo retorna para o início do loop, desta vez, com o contador valendo 2. Esse fluxo irá se repetir até que a condição do loop não seja mais atendida, ou seja, quando o contador for igual a 101.

Vamos ver como escrever esse fluxograma em Portugol:

```
programa
{
    funcao inicio()
    {
        para (inteiro contador=1;contador<=100; contador++)
        {
            se(contador%10==0) {
                escreva(contador, " - Múltiplo de 10\n")
            }
            senao
            {
                escreva(contador,"\n")
            }
        }
    }
}
```

Agora o mesmo código em Javascript:

```
var contador=1
for (var contador=1; contador<=100; contador++) {
    if(contador%10==0) {
        console.log(contador, " - Múltiplo de 10")
    }
    else
    {
        console.log(contador)
    }
}
```

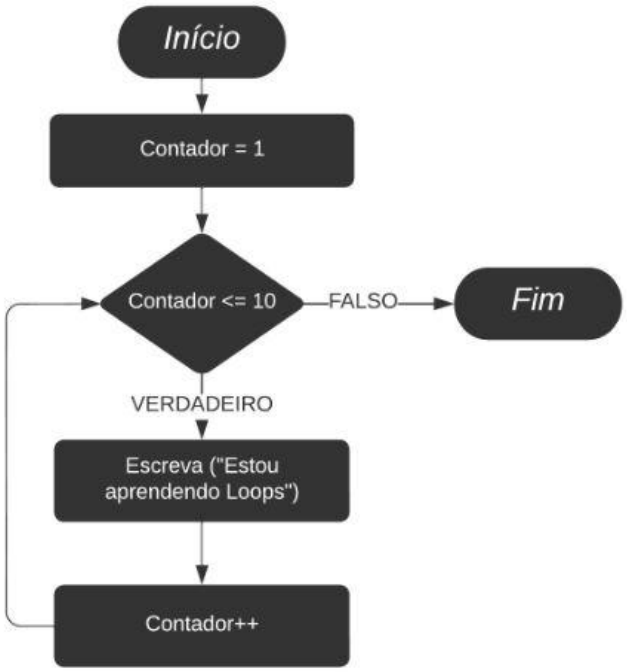
E o resultado no console será algo como:

```
>_ Console   Mensagens

1
2
3
4
5
6
7
8
9
10 - Múltiplo de 10
11
12
13
14
15
16
17
18
19
20 - Múltiplo de 10
21
```

Situação Problema 2:

Vamos criar um loop for, que irá imprimir no console a seguinte frase "**Estou aprendendo Loops**" 10 vezes. Primeiro veremos a resolução através de um fluxograma e depois em Javascript.



Para a criação desse loop, primeiro iniciamos com a palavra reservada **for**. Depois abrimos os parênteses da estrutura e logo em seguida já criamos a nossa variável **i** e damos o valor inicial de 1 à ela. Lembrando que a primeira parte do for será a nossa condição de início, ou seja, de onde queremos que o loop inicie as repetições.

Após isso, precisamos criar a nossa condição de repetição. Nesse caso, queremos a repetição 10 vezes. Na última parte do loop, utilizamos o operador **"++"** (incremento) para que ele modifique o valor da variável **"i"** a cada repetição.

E por último, dentro das chaves do nosso loop for, temos a instrução que queremos repetir.

index.js

```
for(var i = 1; i <=10; i++){
  console.log("Estou aprendendo Loops")
}
```

O resultado desse loop no console, ficará assim:

Exercícios

1. Vamos desenvolver um loop for que calcule a tabuada de multiplicação do 7 e nos mostre como resultado a expressão seguido do resultado, dessa forma:
   7 x 1 = 7
   7 x 2 = 14
   7 x 3 = 21
   7 x 4 = 28
   7 x 5 = 35
   7 x 6 = 42
   7 x 7 = 49
   7 x 8 = 56
   7 x 9 = 63
   7 x 10 = 70

2. Em uma atividade da escola, a professora pediu a seus alunos que dissessem os números pares que existem entre 1 e 20. Para resolver esse problema, podemos criar um loop for e utilizar uma estrutura condicional para verificar os números pares.
   Vale lembrar que para um número ser par, precisa que o resto da sua divisão por dois, seja zero!

3. Crie um loop que execute 15 repetições e cada ciclo desse laço deve imprimir o valor da variável contadora, que nesse caso irá começar no 1.
   Por exemplo, o resultado no terminal precisa ser esse:
   1
   2
   3
   4... e assim por diante.

4. Precisamos de um loop que irá percorrer os números de 1 a 30 e deverá imprimir a frase **"O número x é ímpar"** onde o x na frase deve ser o valor da variável contadora caso ela seja ímpar.
   Por exemplo:
   O número 1 é ímpar
   2
   O número 3 é ímpar
   4...
   e assim por diante.

Estruturas de Repetição

While

Como vimos anteriormente, o Javascript tem diversos comandos para criação de loops. Aprendemos mais a fundo o loop for e agora vamos entender um pouco mais sobre a estrutura do comando While.

A palavra While traduzida para o Português significa “enquanto”. Então basicamente o comando While vai repetir determinado trecho de código, **enquanto** uma condição é avaliada como true.

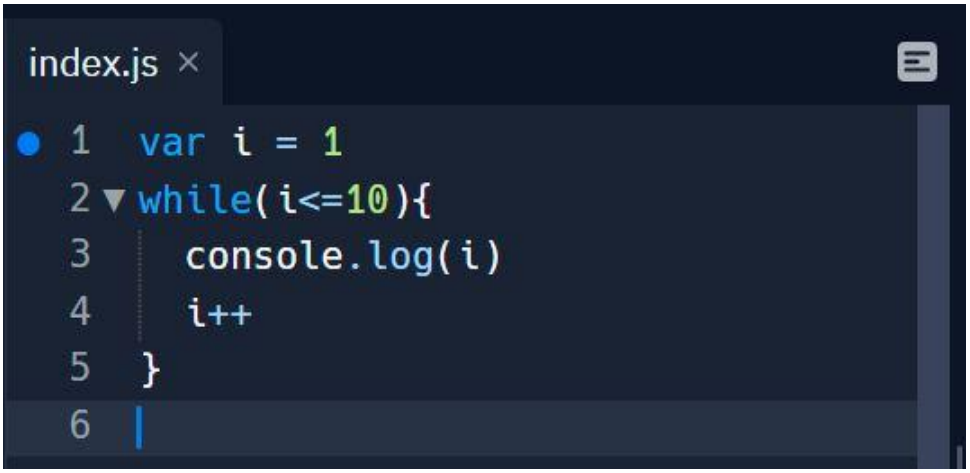
A sintaxe do while:

```
inicialização
while(condição){
   comando A
   comando B
   ...
   incremento
}
```

A estrutura de repetição “while()” funciona com base em uma condição, assim como o “for()”, todavia, as variáveis envolvidas nessa condição precisam ser declaradas e inicializadas antes de chegarmos ao bloco do “while()”.

Outra mudança, refere-se ao incremento da variável que controla o fluxo de repetição, este incremento deve acontecer ao final da execução dos comandos presentes no bloco do “while()”.

Vejamos o exemplo a seguir:
Vamos imprimir os valores de 1 a 10 com o While?

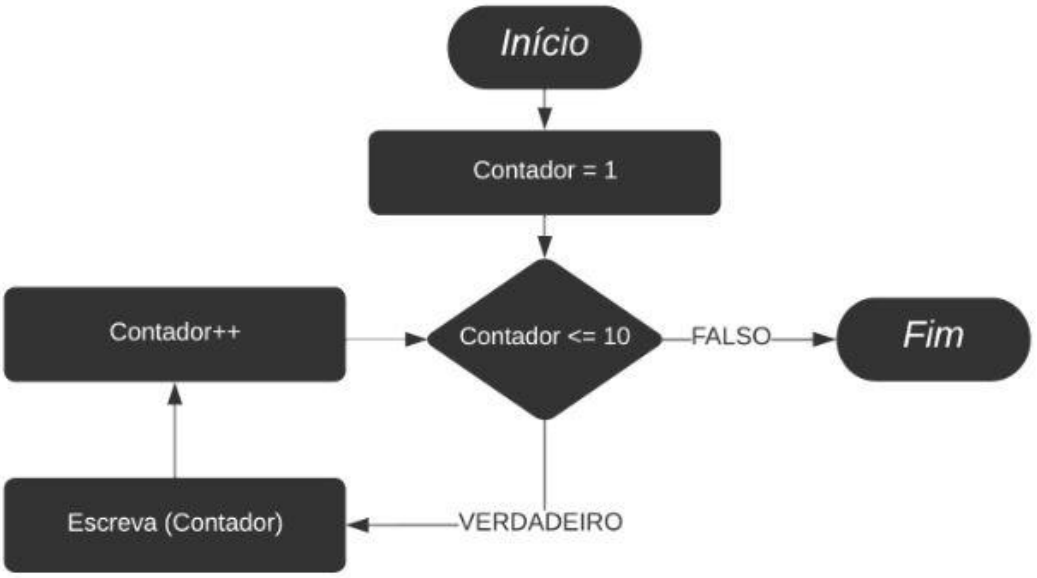


Note que a condição que empregamos ao algoritmo é de que a variável i irá repetir o bloco de código até que o valor dela seja igual a 10. A cada iteração executada é somado à variável **"i"** o valor de 1, utilizando o operador de incremento (i++).

> **Importante!** *Não esqueça de incrementar a variável que controla a repetição, pois caso contrário, seu programa entrará em* ***lopping*** *(repetição infinita) e nunca chegará ao fim.*

Quando a variável **"i"** chegar ao valor de 11, o laço de repetição pára de executar o trecho de código e o resultado do exemplo acima será exibido no console assim:

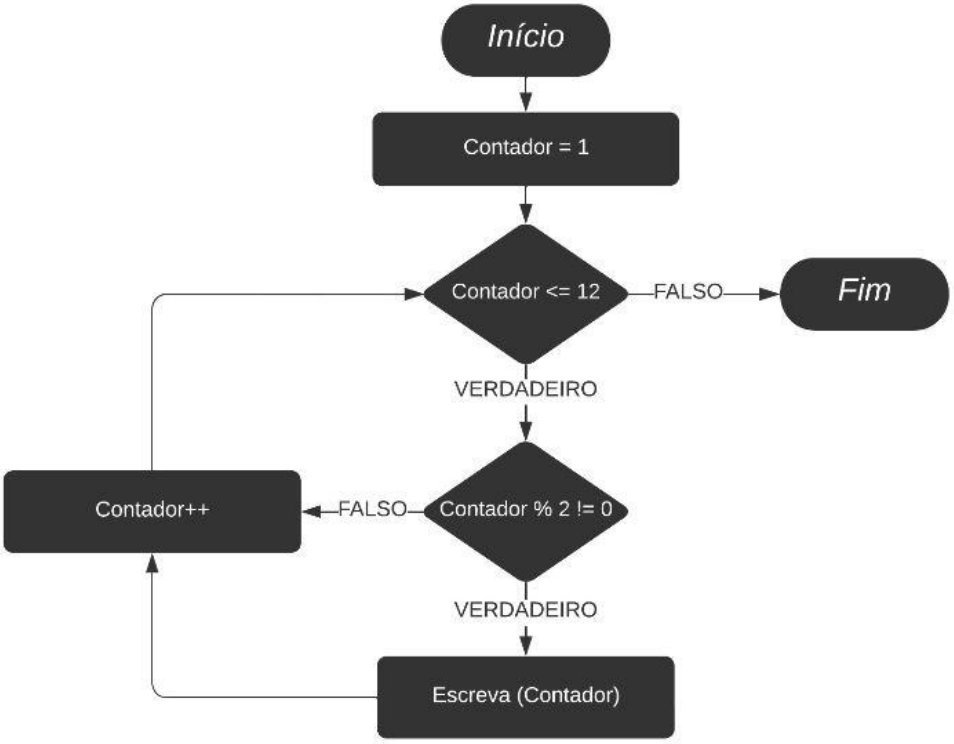
E agora o mesmo ciclo de repetição em fluxograma:

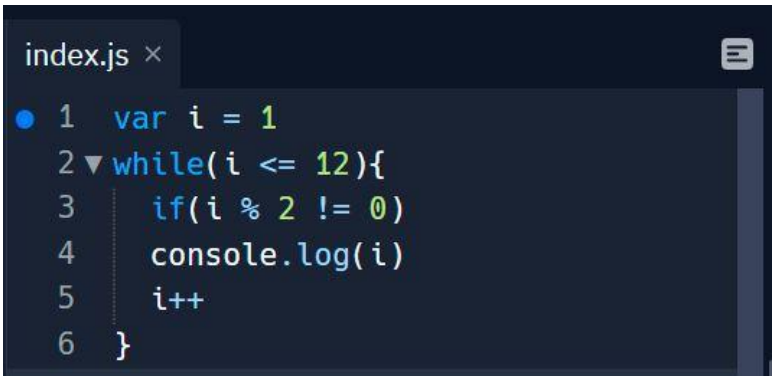


Situação Problema 3:

Vamos criar um laço de repetição While que irá iterar 12 vezes o ciclo e imprimir no console apenas quando o valor dessa iteração for ímpar.

Veremos essa resolução primeiro em fluxograma e depois em Javascript:





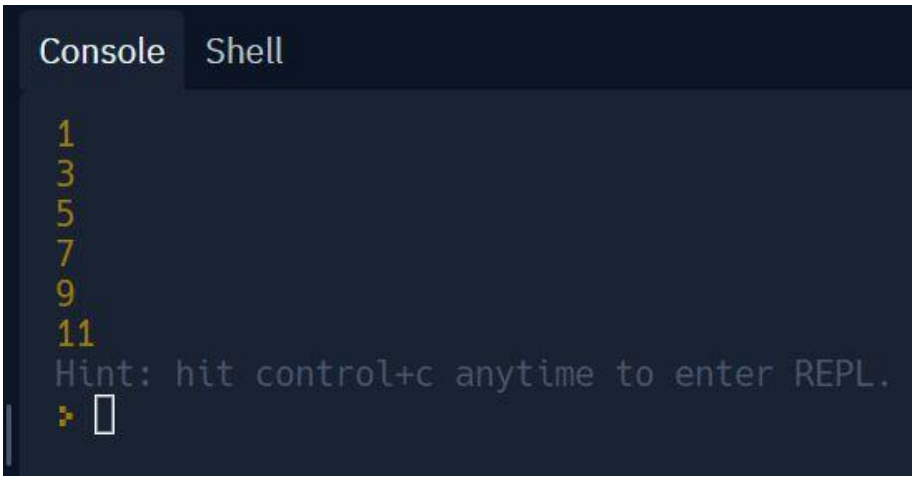
Para essa resolução, criamos a variável **“i” e a inicializamos com o valor 1** que irá nos servir para a validação da condicional.

Depois de declarar o While, criamos a condição dentro dos parênteses, então enquanto o valor de **“i” for menor ou igual a 12**, o trecho de código será repetido.

Nosso trecho de código contém outra estrutura já conhecida por nós, uma condicional **“if”.** Ela está dentro do laço while para poder validar os valores ímpares, utilizando o operador de módulo %. (*Nesse caso, sabemos que um número é ímpar quando o resto de sua divisão por dois é diferente de 0*).

Depois de validarmos os valores ímpares, o **“console.log()”** irá imprimir esses números, e por fim, incrementamos o valor de **“i”** para que a condição seja testada novamente até que em determinado momento retorne “falso”.

O resultado desse código no console ficará assim:



Exercícios

1. Lembra da tabuada que fizemos juntos utilizando o loop for? Agora é a sua vez! Crie a tabuada do 10 utilizando o ciclo de repetição While.

2. Crie um laço de repetição While que irá iterar enquanto o valor de i for menor ou igual a 15 e mostre no console o valor de cada iteração.

3. Faça uma aplicação que leia 10 números do usuário e some somente os valores pares. Ao final, exiba os valores digitados, a relação dos números pares e o valor da soma desses pares.

4. Agora, melhore o código da questão anterior e faça um programa que receba 10 números, calcule e imprima a soma dos números pares e a soma dos números primos.