

Comandos de Repetição

Site: [HackaTruck MakerSpace](#)
Curso: Conceitos e Fundamentos: Algoritmos e Programação
Orientada a Objetos com Swift
Livro: Comandos de Repetição

Impresso por: Matheus Ferreira Santos
Data: terça-feira, 15 ago. 2023, 17:58

Índice

1. Introdução
2. Comando WHILE
3. Comando REPEAT-WHILE
4. Comando FOR

1. Introdução



COMANDOS DE REPETIÇÃO - INTRODUÇÃO

Comandos de repetição são úteis quando desejamos fazer operações repetitivas, sem escrever diversas vezes o mesmo código. Isto se torna ainda mais importante quando não sabemos, na hora em que estamos desenvolvendo, quantas vezes este comando precisará ser repetido. Para solucionar esses empecilhos e aumentar a qualidade do código, vamos entender os comandos **WHILE**, **REPEAT-WHILE** e **FOR**.

Dica

Em nosso EAD utilizamos compiladores online, caso algum deles não carregue, basta clicar em **RUN** .



Caso persista, recarregue a página!

Outras dúvidas ou sugestões entre em contato com contato@hackatruck.com.br.

2. Comando WHILE

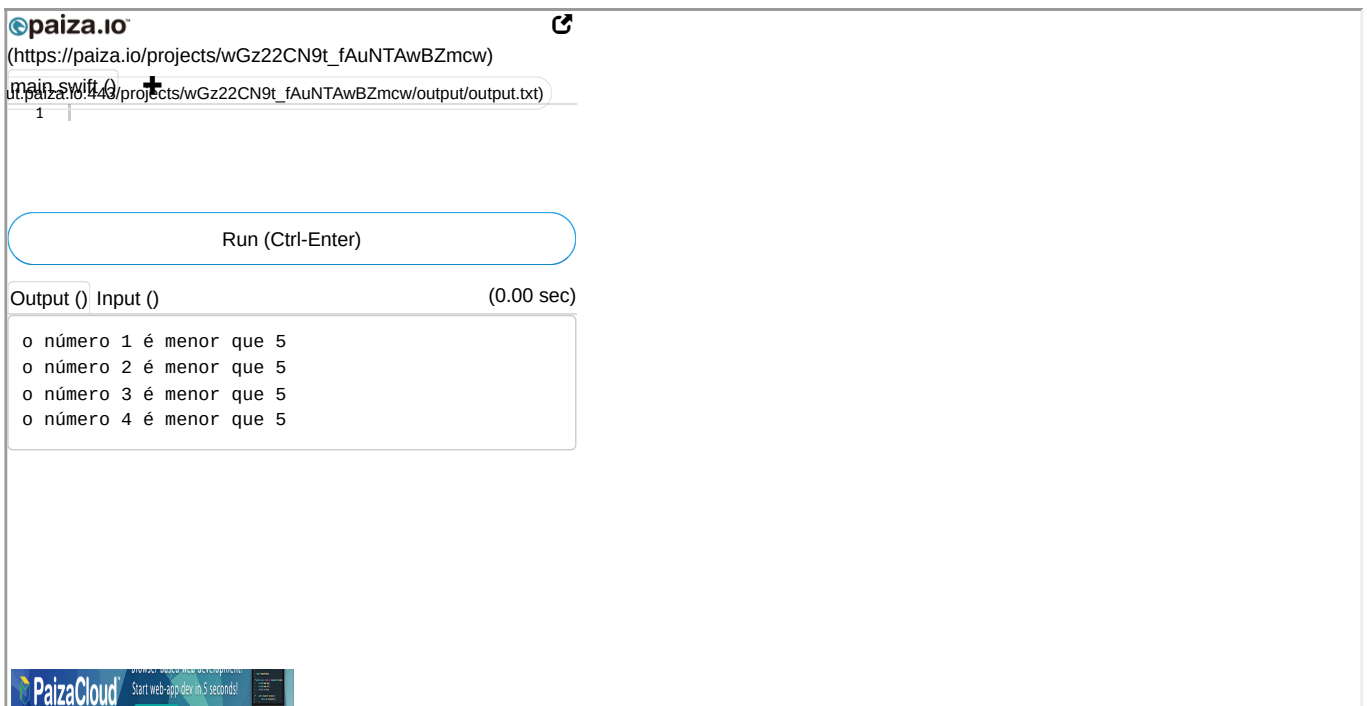


COMANDOS DE REPETIÇÃO - COMANDO WHILE

O comando **while** repete um conjunto de operações enquanto uma condição for verdadeira. Sua sintaxe é a seguinte:

```
while condicao {  
    //Comandos a serem executados em todas repetições enquanto a condição for verdadeira  
}
```

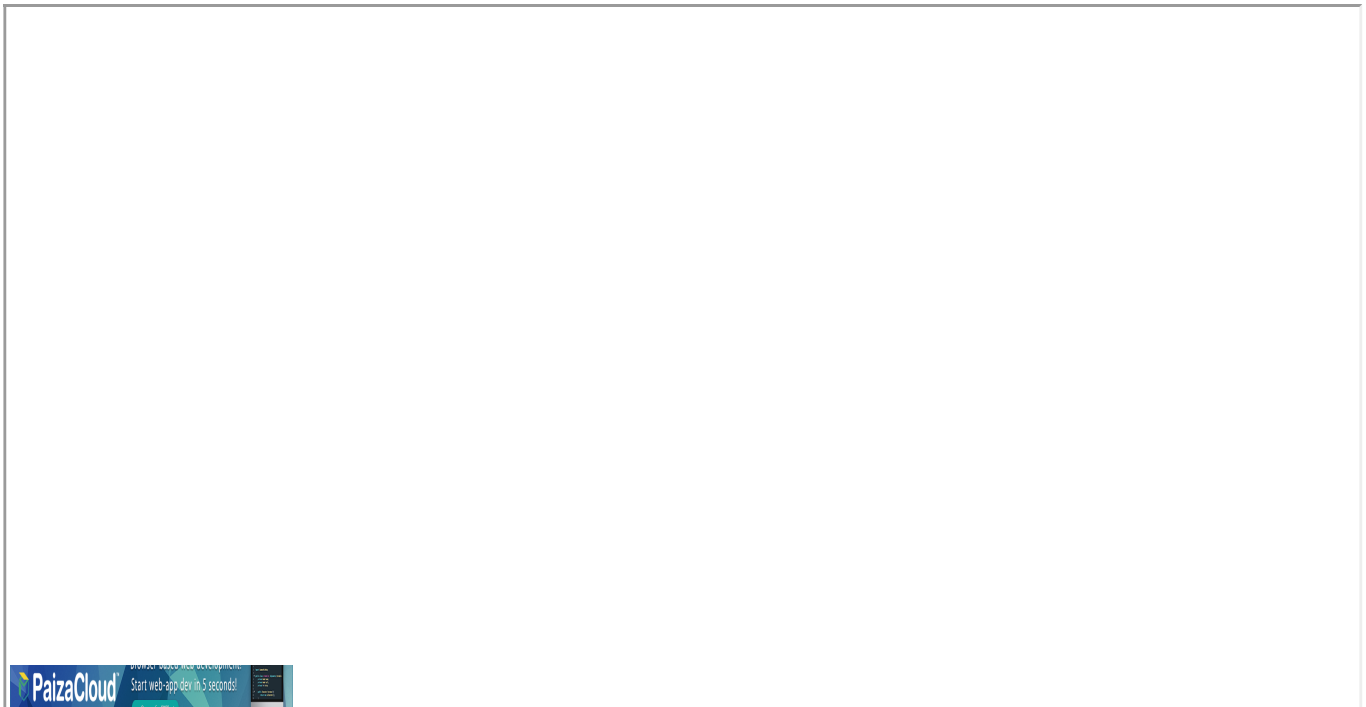
Exemplos:



Ou seja, executamos 4 vezes o comando **print()**, mesmo ele sendo escrito uma única vez, e colocamos nossa condição como **false** utilizando o **else**, fazendo com que nosso laço pare de ser executado assim que a condição passa a ser falsa.

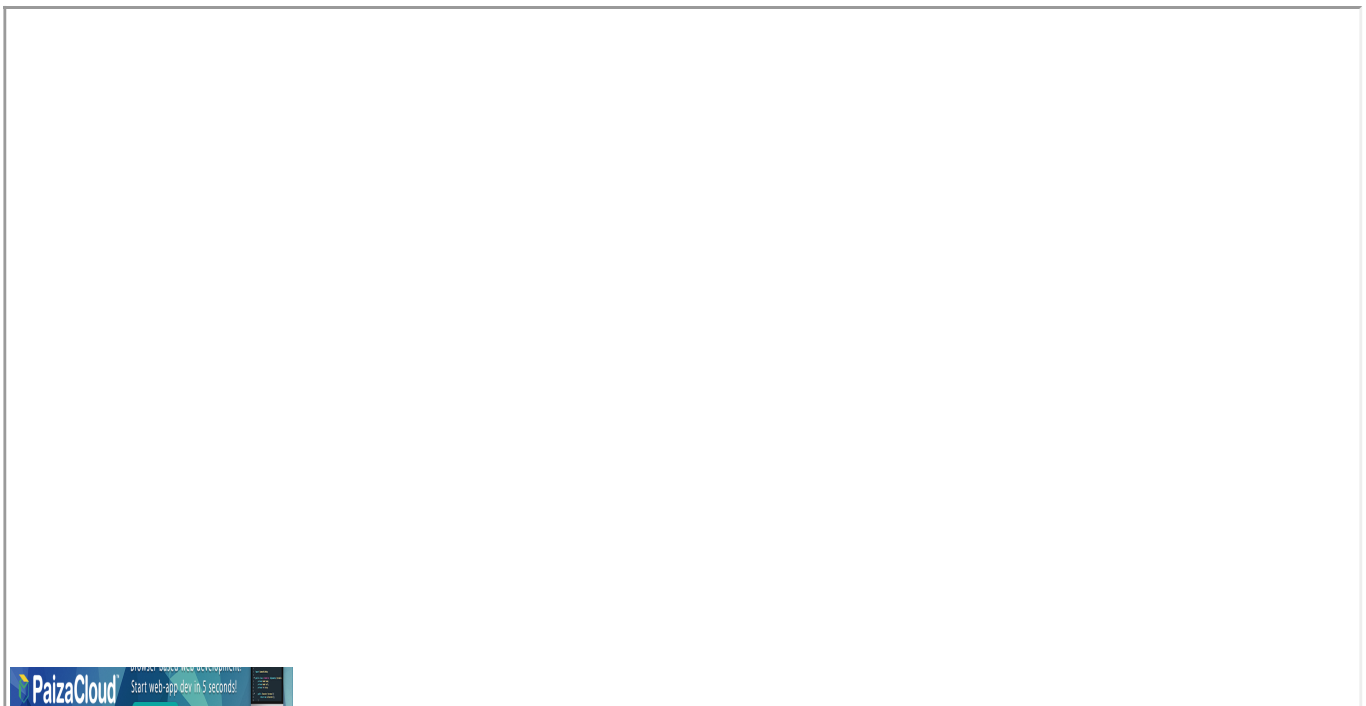
💡 Também podemos utilizar a palavra reservada **break** para parar a execução de um laço.

Vejamos o uso do *break* a partir do mesmo exemplo:



Ou seja, assim que a condição do *if* passa a ser falsa, o comando *break* é executado no *else*, e faz nosso laço parar.

É importante lembrar que, caso nossa condição do *while* seja falsa já no início, **nada será executado**. Por exemplo:



Vamos treinar?

Construa um contador que imprima os números até 512.

paiza.io

(https://paiza.io/projects/Y0upa7QFA5zA2H2a07)

Text

↻

main.swift ()

+

1

Run (Ctrl-Enter)

Output ()

Input ()

(0.00 sec)

PaizaCloud

avoided sleep mode development

Start web-app dev in 5 seconds

Resolução:

paiza.io

(https://paiza.io/projects/cahX95oRCfxdyr0haLDYJw)

main.swift ()

+

1

Run (Ctrl-Enter)

Output ()

Input ()

(0.00 sec)

0

1

2

3

4

5

6

7

8

9

10

11

12

PaizaCloud

avoided sleep mode development

Start web-app dev in 5 seconds

3. Comando REPEAT-WHILE



COMANDOS DE REPETIÇÃO - COMANDO REPEAT-WHILE

O comando **repeat-while** é uma variação do **while** cuja a condição só é verificada após a primeira execução dos comandos nele contidos. Vamos à sintaxe e em seguida, a uma comparação com o **while**:

repeat {

//Comandos a serem executados pelo menos uma vez e posteriormente em todas repetições em que a condição for verdadeira

} while condicao

Comparação:

A screenshot of the Paiza.io online compiler interface. The top bar shows the Paiza.io logo and a share icon. Below it, the URL 'https://paiza.io/projects/K2nZy9h0BUOhFsyl-Qxi5w)' is visible. The code editor shows a Swift file named 'main.swift' with a single line of code: '1 print("Mesmo sem validar a condição será executado ao menos uma vez")'. Below the code editor is a 'Run (Ctrl-Enter)' button. Underneath the button, it says 'Output () Input () (0.00 sec)'. The output area displays the text 'Mesmo sem validar a condição será executado ao menos uma vez'. At the bottom of the interface, there is a PaizaCloud logo and a message 'Start web-app dev in 5 seconds!'.

```
1 print("Mesmo sem validar a condição será executado ao menos uma vez")
```

Com a mesma condição, o nosso **repeat-while** imprimiu sua mensagem enquanto o **while** não.

Vamos ver mais um exemplo de uso:



Vimos que no primeiro exemplo tivemos 5 mensagens impressas no primeiro bloco, com uma condição válida. Já o segundo exemplo está com uma condição falsa, mas ainda assim, imprime a sua primeira mensagem. Essa estrutura pode vir a ser útil dependendo do que estiver fazendo e do cenário que se está trabalhando.

4. Comando FOR



COMANDOS DE REPETIÇÃO - COMANDO FOR

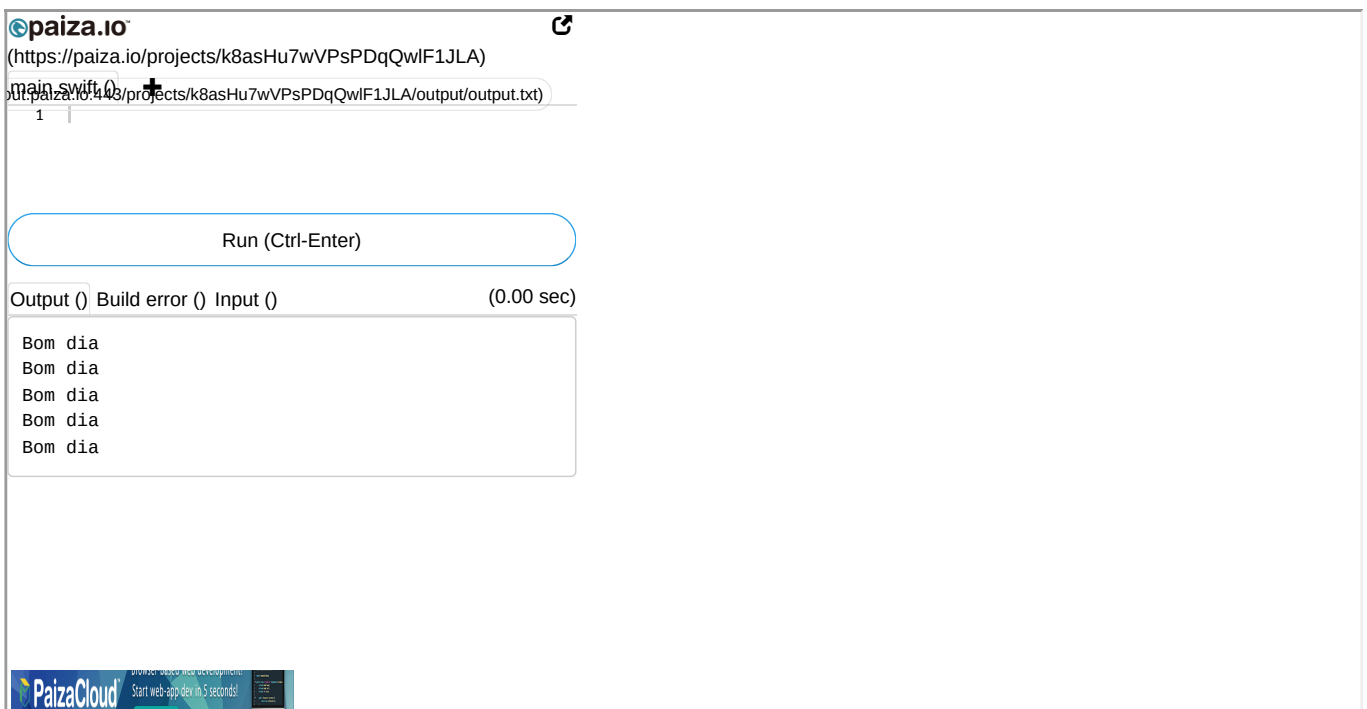
Como vimos anteriormente, esse capítulo nos ensina a não repetirmos muitas vezes a mesma ação de forma manual, correto? Podemos, então, criar laços de repetição (*loops*) que irão trabalhar por nós (maravilhoso, não?), vamos conhecer agora o comando *for*. Vejamos a sintaxe:

For variavel in contador {

//Comando a ser executado em todas repetições até que acabe o contador definido

}

E agora um exemplo básico para começarmos a nos familiarizar com o **for**.

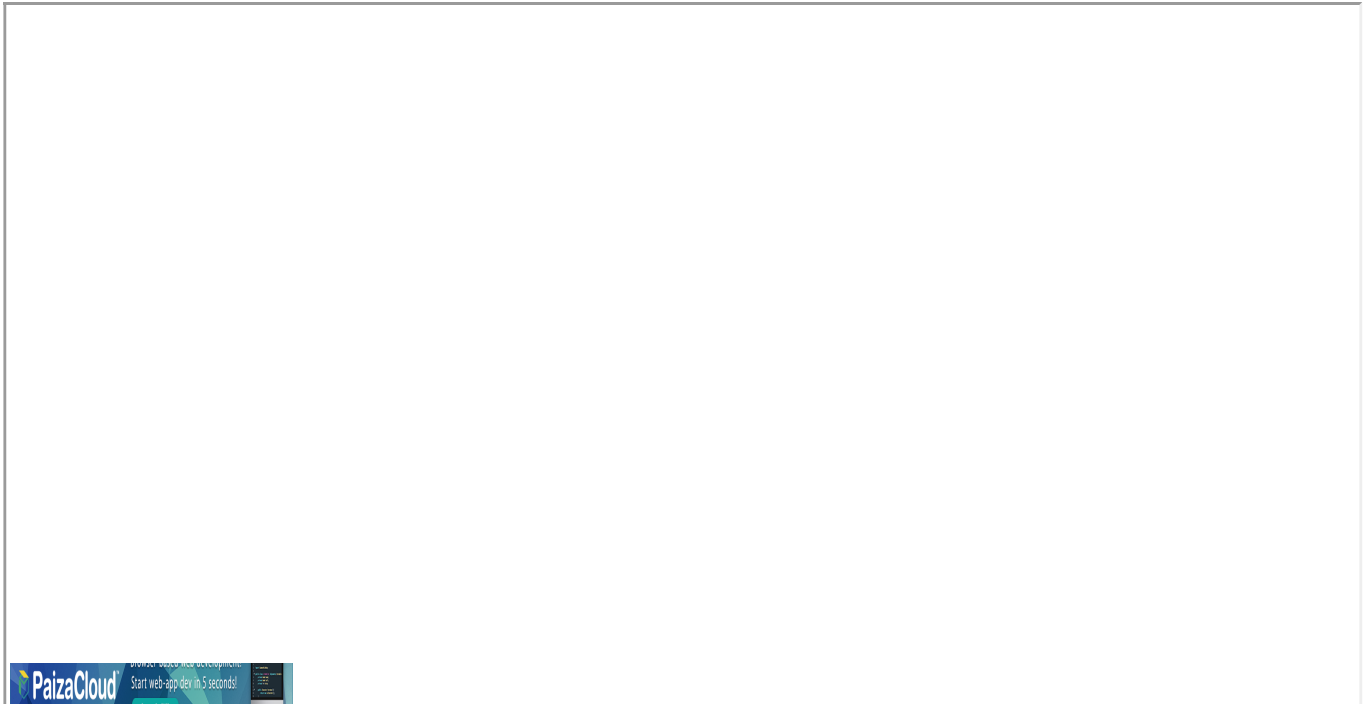


O “Bom dia” foi impresso 5 vezes porque nosso contador é um intervalo de 1 até 5.

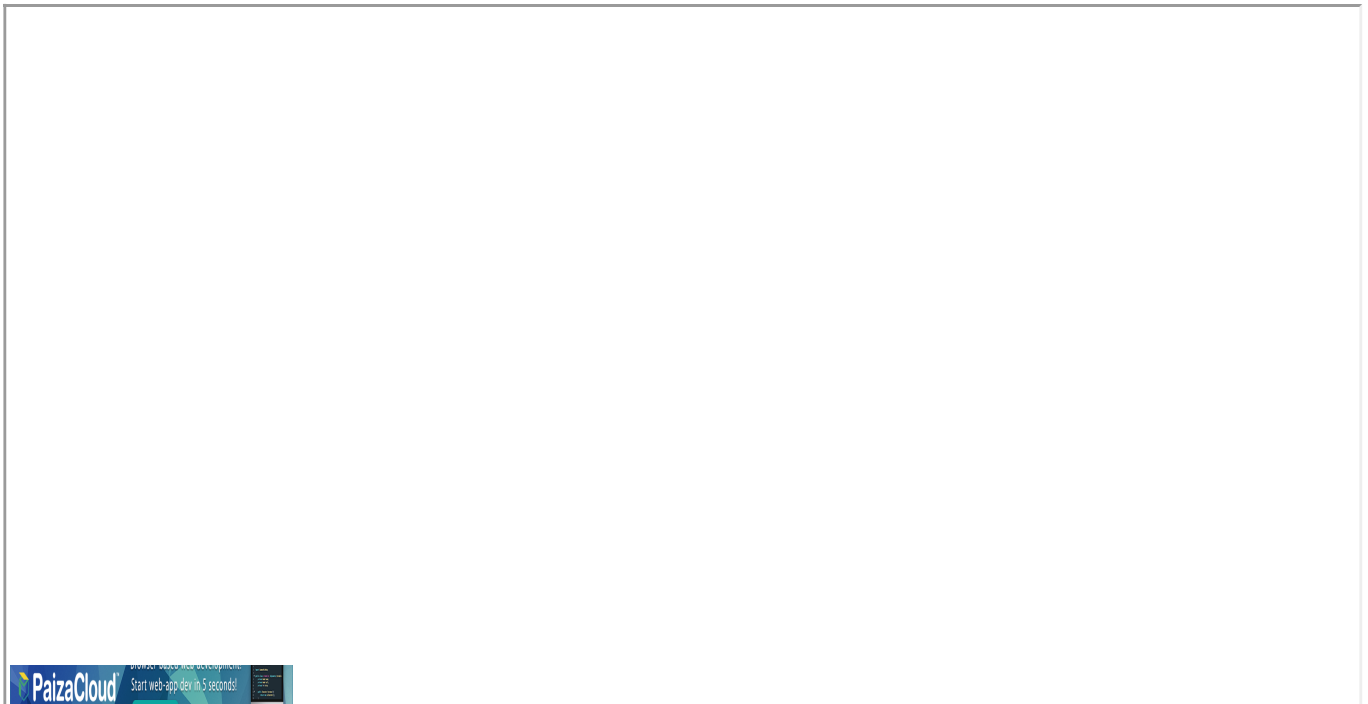
💡 **Aprendemos sobre os operadores de limite no capítulo de Controle de Fluxo e os mesmos serão utilizados aqui. Recapitulando:**

Operador	Operação
A ..< B	É utilizado para definir um intervalo entre um numero A e B excluindo B.
A ... B	É utilizado para definir um intervalo entre um numero A e B incluindo B.

Vamos avançar mais um pouco e imaginar que precisamos imprimir tabuadas. Uma das possibilidades seria gerar 10 **prints()**, correto?



E de fato funcionaria, mas vamos aprender uma nova forma de fazer isso? Para que seja mais eficiente e processado pela máquina solucionando as linhas repetitivas que foram criadas de forma manual, utilizaremos o **for**.



Explicando, **i** é criada em tempo de execução, ou seja, não precisamos declará-la manualmente e ela recebe um valor a cada vez que passamos por essa linha, ou seja:

- Da primeira vez ela tem o valor 1, pois nosso limite é de 1...10
- Da segunda vez ela tem o valor 2, pois já executou os comandos para o valor 1 e assumiu o próximo valor do limite, e assim sucessivamente até que...
- Na última vez ela assume o valor 10, executa todos os comandos e sai desse laço de repetição.

Vamos agora entender isso da ótica dos nossos comandos:


A cada vez que entramos na fase de executar os comandos, a nossa constante *i* está com um novo valor, correto? Então é só combinar esse valor com o que queremos fazer e conseguimos partir de um cenário de 10 linhas de `print()` pra um novo, com apenas 3 linhas!

Vamos olhar de novo e escrever mais algumas tabuadas:

Escrevam a tabuada do 1 ao 10, abaixo dessa:



Já pensou na possibilidade de não fazer o trabalho de forma manual?



Podemos criar um **for** dentro de outro **for** e diminuir ainda mais a escrita de código, só com a ressalva de que precisamos trocar o nome do nosso iterador, normalmente utilizamos *i, j, k* para referenciar nossos iteradores nos laços, mas nada impede que você atribua algum outro nome.

Faça já os exercícios desde capítulo.

Exercícios Comandos de Repetição 

