

ESTRUTURAS DE REPETIÇÃO

Neste tópico aprenderemos a utilizar as estruturas de *loop*. Estruturas estas que nos permitem realizar uma mesma operação uma determinada quantidade de vezes, a qual é controlada por uma variável de controle booleana.

Esse tipo de estrutura é muito utilizada para a manipulação de listas e mapas, devido à estrutura em que eles operam. Porém esse não é seu único uso, apesar de ser o mais comum.

Outro aspecto importante a se ressaltar sobre as estruturas de *loop* é justamente o fator que as torna o que são, os loops (também conhecidos com repetições ou voltas). Uma estrutura de *loop* possui uma condição de parada, e até que essa condição seja atendida, ela não cessa sua execução, o que pode levar a uma execução infinita, também conhecido como *loop* eterno.

Estrutura de loop While

O *While* é uma estrutura de loop similar ao *IF*. Da mesma forma, possui uma condição para executar um bloco de comandos. Porém se distingue dele pelo fato de executar o bloco de comandos até que a condição imposta torna-se falsa. Até esse momento chegar, o bloco de comandos previsto é executado.

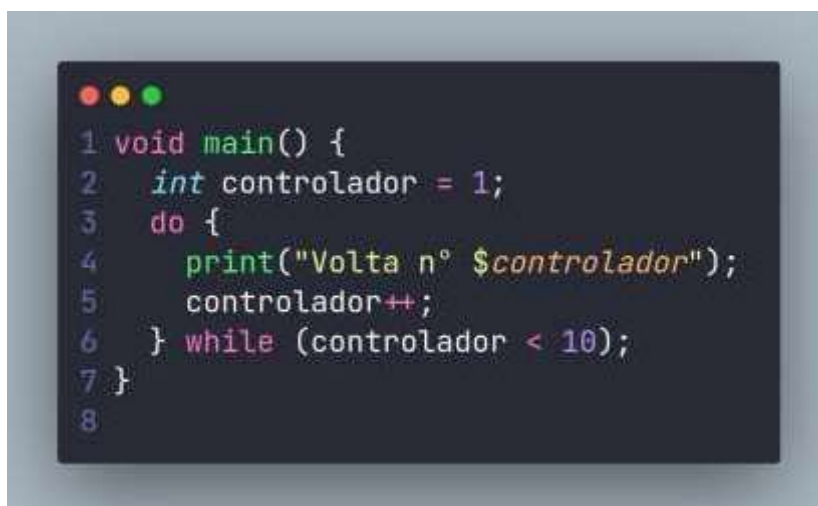


```
1 void main() {  
2   int controlador = 1;  
3   while (controlador < 10) {  
4     print("Volta nº $controlador");  
5     controlador++;  
6   }  
7 }
```

Talvez você tenha notado algo incomum em nosso comando “*print*” usual. Foi introduzido um símbolo de cifrão (\$). Isso informa ao Dart que desejamos combinar uma *String* literal ao valor de uma variável ou constante. Muito útil quando desejamos imprimir texto juntamente com referências de variáveis e constantes.

Estrutura de loop Do while

A estrutura de loop “*Do While*” é praticamente igual a estrutura *While*, a grande diferença se encontra no modo como a função valida a condição de parada do loop. Enquanto o *While* valida a condição antes de iniciar o loop, o “*Do While*” o faz ao final da volta. Isso possibilita a execução do loop ao menos uma vez, dando a oportunidade de inicializar o controlador do loop após o início do mesmo. É importante ressaltar que esse tipo de implementação pode ocasionar problemas na execução do *loop*, como por exemplo torná-lo eterno, ou seja, onde a condição de parada nunca é alcançada.



```
1 void main() {  
2     int controlador = 1;  
3     do {  
4         print("Volta nº $controlador");  
5         controlador++;  
6     } while (controlador < 10);  
7 }  
8
```

Estrutura de loop For

O *For* é uma estrutura de controle que estabelece um *loop* de repetição baseado em um contador numérico, diferenciando-se dos demais pelo fato de não utilizar uma condição booleana externa, já que a estrutura da sua sintaxe prove tudo que essa função necessita ser executada, impedindo de ações externas ao loop afetem a execução do mesmo.

Outro fator importante a se levar em consideração é que essa estrutura se baseia em uma contagem e não simplesmente em um valor booleano. Isso quer dizer que a repetição será executada um número limitado de vezes, devido a sua estrutura interna de contagem (incremento ou decremento).

```
1 void main() {
2     for (int indice = 1; indice < 10; indice++) {
3         print("Volta n° $indice");
4     }
5 }
```

Estrutura for-in

A estrutura de *loop for-in* tem um nome muito parecido com o *For*, porém são estruturas distintas. Enquanto o *loop For* utiliza um contador para gerenciar as voltas do *loop*, o *for-in* faz uso de um contador intrínseco para gerenciar as voltas do loop baseada nas posições de uma coleção (Listas, sets e mapas). Isso faz dele uma estrutura de *loop* destinada para interações em coleções.

```
1 void main() {
2     const List numeros_primos = [2, 3, 5, 7, 11]; // ...
3     for (int num in numeros_primos) {
4         print(num);
5     }
6 }
7
```

Resultado:

2

3

5

7

11