IF ELSE

if-else é uma estrutura condicional. Para fazer isso em Kt:

1. Crie uma variável numeral
2. Certifique-se do numero ser maior que zero
3. É simples, se for verdadeiro execute dentro do if e se for falso execute dentro do else

```
val numero = 10
if (numero > 0) {
   println("Número positivo")
} else {
   println("Número negativo ou zero")
}
```

WHEN

when é uma alternativa mais limpa ao switch em kt (em relação ao java). Ele compara uma variável com várias opções possíveis e executa o bloco correspondente.

1. defina uma variável (por exemplo, uma opção de menu).
2. use when para comparar a variável com diferentes valores possíveis.
3. especifique o que deve acontecer para cada valor.

```
val opcao = 2
when (opcao) {
   1 -> println("Opção 1 escolhida")
   2 -> println("Opção 2 escolhida")
   3 -> println("Opção 3 escolhida")
   else -> println("Opção inválida")
}
```

FOR

O loop for é usado para repetir um bloco de código um número específico de vezes ou para percorrer uma coleção.

1. crie um intervalo de números ou uma coleção de itens (array).
2. use o loop for para percorrer esse intervalo ou coleção.
3. para cada item, execute o que for necessário dentro do loop.

```
for (i in 1..5) {
   println("Contagem: $i")
}
```

Exibir itens da array

Se você quiser percorrer e exibir todos os itens de uma array, pode usar o loop for para acessar cada elemento.

1. Crie a array com vários elementos.
2. Use um loop for para iterar por todos os itens do array.
3. Imprima cada item durante o loop.

```
val lista = arrayOf(10, 20, 30, 40, 50)
for (item in lista) {
   println(item)
}
```

Exibir o 6º Elemento da Array

Arrays em kt começam com [0], então o 6º elemento estará no [5]. Para acessá-lo, basta referenciar o índice desejado.

1. Crie uma array com vários elementos.
2. Acesse o 6º elemento usando [5].
3. Imprima o valor do 6º elemento.

```
val lista = arrayOf(10, 20, 30, 40, 50, 60, 70, 80)
println("O 6º elemento da array é: ${lista[5]}")
```

WHILE

O while é útil quando você precisa repetir uma ação enquanto uma condição for verdadeira. O código dentro do loop while é executado até que a condição se torne falsa.

1. Defina um contador.
2. Use while para verificar a condição que controla o loop.
3. A cada iteração, modifique a variável de controle para que o loop possa eventualmente terminar.

```
var contador = 0
while (contador < 5) {
   println("While loop contador: $contador")
   contador++
}
```

Soma

1. Defina uma função soma que aceita parâmetros do tipo Int.
2. Dentro da função, retorne a soma desses dois números.

```
fun soma(a: Int, b: Int): Int {
   return a + b
}
```

Subtração

1. Defina uma função subtracao com dois parâmetros a e b.
2. Dentro da função, retorne a subtração de b de a.

```
fun subtracao(a: Int, b: Int): Int {
   return a - b
}
```

Multiplicação

1. Defina uma função chamada multiplicacao com dois parâmetros a e b.
2. Retorne o produto de a e b dentro da função.

```
fun multiplicacao(a: Int, b: Int): Int {
    return a * b
}
```

10. Divisão

1. Defina uma função chamada divisao com dois parâmetros a e b.
2. Verifique se b é zero antes de realizar a divisão.
3. Se b for zero, exiba uma mensagem de erro. Caso contrário, retorne o resultado da divisão.

```
fun divisao(a: Int, b: Int): Int {
    if (b == 0) {
        println("Erro: divisão por zero")
        return 0
    }
    return a / b
}
```