A declaração de ***array*** no **Kotlin** é bem semelhante a instanciar um novo objeto, basta declarar um **array com o tipo dos elementos e o tamanho que ele vai ter**. A **atribuição de valor no índice** é igual a outras linguagens. Também da pra iniciar o array de forma mais enxuta usando um método que já recebe valores do array:

```
val arrayInteiro = IntArray( size: 4)

arrayInteiro[0] = 1
arrayInteiro[1] = 2
arrayInteiro[2] = 3
arrayInteiro[3] = 4
```

```
val arrayNumero = intArrayOf(16,66,35,41,23)
```

Da pra **percorrer um Array** usando um **loop for** ou usando um **for each** ***(o for each permite executar uma fução para cada elemento da lista, é mto semelhante a lambda expressios do Java):***

```
var maiorNumero = 0

for (numero in arrayNumero){
    if(numero > maiorNumero){
        maiorNumero = numero
    }
}
```

```
var menorNumero = Int.MAX_VALUE
arrayNumero.forEach { numero -> if(numero < menorNumero){
    menorNumero = numero} }
```

As vezes temos que **fazer modificações em valores de um Array**, o **Kotlin** oferece duas opções uteis para esse propósito, que nos **"livra"** de ter que incrementar a posição do **Array** manualmente.

No **"for" iterado pela propriedade *indices* do Array o Kotlin já fornece o range de iteração necessário**, já no **forEachIndexed ele funciona como um lambda expression**, então **podemos fazer coisas mais poderosas** como **fornecer o indice do array e o valor que será alterado:**

```
for(indice in arrayNumero.indices){
    arrayNumero[indice] = arrayNumero[indice] * 2
}
```

```
arrayNumero.forEachIndexed { indice, numero ->
    arrayNumero[indice] = numero * 2
}
```

Obviamente existem **funções de agregação** nos **Arrays**, como ***min(), max(), average()*** etc..., além de funções de filtro como ***filter(), all(), any()*** etc...

```
val maiorNumero = arrayNumero.maxOrNull()

val menorNumero = arrayNumero.minOrNull()

val media = arrayNumero.average()
```

```
arrayNumero.all { it >= 5 }
arrayNumero.any { it >= 5}
arrayNumero.filter { numero -> numero >= 5 }
```