1. Tipe Fungsi di Kotlin

Tipe fungsi dalam Kotlin memungkinkan kita untuk merepresentasikan fungsi sebagai tipe data. Ini berarti kita bisa menyimpan fungsi dalam variabel, melewatkan fungsi sebagai parameter, atau mengembalikan fungsi dari fungsi lain.

Deklarasi Tipe Fungsi

Tipe fungsi di Kotlin dideklarasikan dengan menggunakan notasi:

```
(val|var) namaFungsi: (TipeParameter1, TipeParameter2, ...) -> TipePengembalian
```

Contoh:

```
val sum: (Int, Int) -> Int = { a, b -> a + b }
```

Fungsi `sum` adalah tipe fungsi yang menerima dua parameter `Int` dan mengembalikan nilai `Int`.

Contoh Tipe Fungsi dalam Fungsi

```
fun operate(a: Int, b: Int, operation: (Int, Int) -> Int): Int {
    return operation(a, b)
}

fun main() {
    val result = operate(5, 3) { x, y -> x + y }
    println(result)  // Output: 8
}
```

Pada contoh di atas, `operation` adalah parameter yang berupa tipe fungsi `(Int, Int) -> Int`.

---

2. Lambda di Kotlin

Lambda adalah fungsi anonim (tanpa nama) yang bisa digunakan sebagai ekspresi. Lambda digunakan untuk membuat fungsi kecil dengan cara yang ringkas, biasanya dalam konteks fungsi tingkat tinggi (higher-order functions).

Sintaks Lambda

Lambda ditulis dalam bentuk:

```
{ parameter1, parameter2 -> ekspresi }
```

Contoh lambda sederhana:

```
val multiply = { x: Int, y: Int -> x * y }
println(multiply(3, 4))  // Output: 12
```

Lambda bisa digunakan secara langsung sebagai parameter fungsi:

```
fun calculate(a: Int, b: Int, operation: (Int, Int) -> Int): Int {
    return operation(a, b)
}

fun main() {
    val result = calculate(5, 3) { x, y -> x * y }
    println(result)  // Output: 15
}
```

---

3. Higher-Order Functions (Fungsi Tingkat Tinggi)

Fungsi tingkat tinggi adalah fungsi yang bisa menerima fungsi lain sebagai parameter, atau mengembalikan fungsi. Ini adalah fitur penting yang membuat Kotlin menjadi sangat fleksibel dan mendukung pemrograman fungsional.

Contoh Higher-Order Function

```
fun operate(a: Int, b: Int, operation: (Int, Int) -> Int): Int {
    return operation(a, b)
}

fun add(x: Int, y: Int): Int = x + y

fun main() {
    // Menggunakan fungsi sebagai parameter
    val result1 = operate(4, 2, ::add)
    println(result1)  // Output: 6

    // Menggunakan lambda sebagai parameter
    val result2 = operate(4, 2) { x, y -> x * y }
    println(result2)  // Output: 8
}
```

---

4. Lambda dengan Parameter Tunggal

Jika lambda hanya memiliki satu parameter, Anda bisa menggunakan kata kunci `it` sebagai pengganti mendeklarasikan nama parameter.

```
val square = { it: Int -> it * it }
println(square(5))  // Output: 25
```

`it` secara otomatis mewakili satu-satunya parameter dari lambda.

---

5. Anonymous Function

Anonymous function dalam Kotlin mirip dengan lambda, tetapi memungkinkan kita menentukan tipe pengembalian secara eksplisit dan mendukung `return` di dalam fungsi.

```
val sum = fun(a: Int, b: Int): Int {
    return a + b
}

fun main() {
    println(sum(2, 3))  // Output: 5
}
```

Anonymous function memungkinkan lebih banyak kontrol dibandingkan lambda.

---

6. Inline Functions

Inline functions digunakan untuk mengurangi overhead kinerja yang disebabkan oleh pembuatan objek lambda. Fungsi inline akan menggabungkan kode lambda langsung ke tempat di mana fungsi tersebut dipanggil.

Contoh Inline Function

```
inline fun operate(a: Int, b: Int, operation: (Int, Int) -> Int): Int {
    return operation(a, b)
}

fun main() {
    val result = operate(4, 5) { x, y -> x * y }
    println(result)  // Output: 20
}
```

Dengan kata kunci `inline`, Kotlin tidak akan membuat objek lambda baru untuk setiap pemanggilan, melainkan memasukkan kode lambda langsung ke dalam fungsi.

---

7. Ekspresi Lambda pada Koleksi

Kotlin menyediakan berbagai fungsi tingkat tinggi seperti `map`, `filter`, dan `reduce` untuk bekerja dengan koleksi. Fungsi-fungsi ini menerima lambda sebagai parameter untuk mengolah data.

Contoh Penggunaan Lambda pada Koleksi

```
fun main() {
    val numbers = listOf(1, 2, 3, 4, 5)

    // Menggunakan map untuk mengubah nilai elemen
```

```
    val doubled = numbers.map { it * 2 }
    println(doubled)  // Output: [2, 4, 6, 8, 10]

    // Menggunakan filter untuk memfilter elemen
    val evenNumbers = numbers.filter { it % 2 == 0 }
    println(evenNumbers)  // Output: [2, 4]
}
```

Kesimpulan

- **Tipe fungsi** di Kotlin memungkinkan kita menyimpan fungsi dalam variabel dan menggunakannya sebagai parameter atau nilai balik dari fungsi lain.
- **Lambda** adalah cara ringkas untuk mendefinisikan fungsi anonim. Lambda sering digunakan dalam fungsi tingkat tinggi seperti `map`, `filter`, dan `reduce`.
- **Higher-order functions** memungkinkan Anda menggunakan fungsi sebagai parameter atau mengembalikan fungsi.
- **Inline functions** digunakan untuk meningkatkan kinerja dengan mencegah pembuatan objek lambda.