Higher - Order Functions

Example 1 : Apply Function

A simple higher-order function that applies a function to a value:

$$[X, Y]$$
$$apply : (X \longrightarrow Y) \longrightarrow (X \longrightarrow Y)$$
$$\forall f : X \longrightarrow Y \bullet \forall x : X \bullet apply(f)(x) = f(x)$$

apply takes a function and a value, returns the result.

Example 2 : Using Apply

$$square : \mathbb{N} \longrightarrow \mathbb{N}$$
$$result : \mathbb{N}$$
$$\forall n : \mathbb{N} \bullet square(n) = n * n$$
$$result = apply(square)(5)$$

$result = 25$

Example 3 : Compose Function

Function composition as a higher-order operation:

$$[A, B, C]$$
$$compose : (B \longrightarrow C) \times (A \longrightarrow B) \longrightarrow (A \longrightarrow C)$$
$$\forall f : B \longrightarrow C \bullet \forall g : A \longrightarrow B \bullet \forall x : A \bullet compose(f, g)(x) = f(g(x))$$

compose takes two functions and returns their composition.

Example 4 : Using Compose

$$addOne : \mathbb{N} \longrightarrow \mathbb{N}$$
$$double : \mathbb{N} \longrightarrow \mathbb{N}$$
$$addOneAndDouble : \mathbb{N} \longrightarrow \mathbb{N}$$
$$\forall n : \mathbb{N} \bullet addOne(n) = n + 1$$
$$\forall n : \mathbb{N} \bullet double(n) = 2 * n$$
$$addOneAndDouble = compose(double, addOne)$$

addOneAndDouble(5) = double(addOne(5)) = double(6) = 12

Example 5 : Twice Function

A function that applies another function twice:

$$[X]$$
$$twice : (X \longrightarrow X) \longrightarrow (X \longrightarrow X)$$
$$\forall f : X \longrightarrow X \bullet \forall x : X \bullet twice(f)(x) = f(f(x))$$

twice takes a function and returns a function that applies it twice.

Example 6 : Using Twice

$$
\begin{array}{|l}
increment : \mathbb{N} \longrightarrow \mathbb{N} \\
addTwo : \mathbb{N} \longrightarrow \mathbb{N} \\
\hline
\forall\, n : \mathbb{N} \bullet increment(n) = n + 1 \\
addTwo = twice(increment)
\end{array}
$$

addTwo(5) = increment(increment(5)) = 7

Example 7 : Constant Function

A higher-order function that returns a constant function:

$$
\begin{array}{|l}
[X, Y] \\
\hline
constant : Y \longrightarrow (X \longrightarrow Y) \\
\hline
\forall\, y : Y \bullet \forall\, x : X \bullet constant(y)(x) = y
\end{array}
$$

constant takes a value and returns a function that always returns that value.

Example 8 : Using Lambda with Higher - Order Functions

$$
\begin{array}{|l}
resultLambda : \mathbb{N} \\
\hline
resultLambda = apply(\lambda\, x : \mathbb{N} \bullet x * x)(7)
\end{array}
$$

$resultLambda = 49$

Example 9 : Flip Function

A function that flips the arguments of a binary function:

$$
\begin{array}{|l}
[X, Y, Z] \\
\hline
flip : (X \times Y \longrightarrow \mathbb{Z}) \longrightarrow (Y \times X \longrightarrow \mathbb{Z}) \\
\hline
\forall\, f : X \times Y \longrightarrow \mathbb{Z} \bullet \forall\, x : X \bullet \forall\, y : Y \bullet flip(f)(y, x) = f(x, y)
\end{array}
$$

flip swaps the order of arguments to a function.

Example 10 : Best Practices

When using higher-order functions:

1. Use type parameters to make functions generic
2. Keep function signatures clear and simple
3. Compose small functions to build complex behavior
4. Use lambda expressions for inline function definitions
5. Higher-order functions enable functional programming patterns