

함수형 프로그래밍

함수형 프로그래밍이란?

함수형 프로그래밍(Functional Programming)은 프로그래밍 패러다임 중 하나로, 수학적 함수 개념을 기반으로 하는 프로그래밍 방법입니다. 함수형 프로그래밍에서는 변수의 값을 변경시키지 않고, 함수의 조합으로 문제를 해결합니다. 이를 통해 복잡한 문제를 간단하게 표현할 수 있고, 코드의 가독성과 재사용성이 높아집니다.

- 함수형 프로그래밍의 마크(??)



함수와 값의 관계를 표현하는 수학적 기호입니다. 수학에서는 함수를 표현할 때 $f(x)$ 와 같이 표기합니다. 이 때, f 는 함수의 이름이며, x 는 함수에 인자로 전달되는 값입니다. 이와 비슷하게 함수형 프로그래밍에서는 보통 λ (lambda) 기호를 사용하여 함수를 표현합니다. 이를 람다식(lambda expression)이라고 부릅니다. 람다식은 익명 함수를 만들기 위해 사용됩니다. 예를 들어, 자바스크립트에서는 다음과 같은 코드를 작성할 수 있습니다.

함수형 프로그래밍 특징

◆ **상태 변경을 최소화:** 함수형 프로그래밍에서는 변수의 값을 변경시키지 않습니다. 대신 입력 값에 대한 출력 값을 반환하며, 이로 인해 코드의 가독성과 디버깅이 편리해집니다.

```
// 변수를 변경하지 않는 예제
const numbers = [1, 2, 3, 4, 5];

const sum = numbers.reduce((acc, cur) => acc + cur, 0); // 입력 값에 대한 출력 값을 반환하므로 변수 변경 X
console.log(sum); // 15
```

◆ **순수 함수:** 함수형 프로그래밍에서는 부작용이 없는 순수 함수를 사용합니다. 즉, 같은 입력에 대해서는 항상 같은 출력을 반환하며, 외부의 상태를 변경하지 않습니다. 이를 통해 코드의 테스트가 용이해집니다.

```
// 순수 함수를 사용하는 예제
const add = (a, b) => a + b; // 같은 입력에 대해서는 항상 같은 출력을 반환하므로 순수 함수
let c = 1;

const pureFunction = (a, b) => add(a, b); // 외부의 상태를 변경하지 않으므로 순수 함수
const impureFunction = (a) => (c += a); // 외부의 상태를 변경하므로 순수 함수 X

console.log(pureFunction(1, 2)); // 3
console.log(c); // 1
```

◆ **불변성**: 함수형 프로그래밍에서는 변수가 변경되지 않기 때문에 데이터의 불변성을 보장합니다. 이로 인해 다중 스레드 환경에서 안정적인 코드를 작성할 수 있습니다.

```
// 데이터의 불변성을 보장하는 예제
const numbers = [1, 2, 3, 4, 5];

const doubledNumbers = numbers.map((num) => num * 2); // 새로운 배열을 반환하므로 데이터의 불변성을 보장
console.log(numbers); // [1, 2, 3, 4, 5]
console.log(doubledNumbers); // [2, 4, 6, 8, 10]
```

◆ **함수의 조합**: 함수형 프로그래밍에서는 함수를 조합하여 새로운 함수를 만들어 문제를 해결합니다. 이를 통해 코드의 재사용성이 높아집니다.

```
// 함수를 조합하여 새로운 함수를 만드는 예제
const numbers = [1, 2, 3, 4, 5];

const sum = (numbers) => numbers.reduce((acc, cur) => acc + cur, 0);
const average = (numbers) => sum(numbers) / numbers.length;
const doubledAverage = (numbers) => average(numbers.map((num) => num * 2));

console.log(doubledAverage(numbers)); // 11
```

함수형 프로그래밍의 장점

- 상태 변경이나 부작용이 없는 순수 함수를 사용하기 때문에 코드가 더 간결하고 이해하기 쉽습니다.
- 코드의 재사용성이 높아지며, 유지보수가 쉬워집니다.
- 병렬 처리가 더 용이하며, 멀티스레딩 프로그래밍에서도 안전합니다.
- 함수 단위로 테스트하기 용이하고, 테스트 케이스를 작성하기 쉬워집니다.
- 사이드 이펙트(side effect)가 없으므로 예측 가능한 코드를 작성할 수 있습니다.

함수형 프로그래밍의 단점

- 함수형 프로그래밍 스타일로 구현할 때, 객체 지향 프로그래밍의 상속, 다형성 등을 활용하기 어렵습니다.

- 함수형 프로그래밍 언어들은 일반적으로 C나 자바와 같은 언어보다 성능이 좋지 않은 경향이 있습니다.
- 함수형 프로그래밍에서는 상태 변경이나 입출력 작업이 없기 때문에, 실제 프로그램에서는 상태 변화와 입출력 작업이 필수적인 경우가 많습니다. 이 경우에는 함수형 프로그래밍이 적합하지 않을 수 있습니다.