**1. 함수형 프로그래밍, 절차지향 프로그래밍, 객체지향프로그래밍이란 무엇인가**

**- 함수형 프로그래밍**

함수형 프로그래밍은 자료 처리를 수학적 함수의 계산으로 취급하고,

상태와 가변 데이터를 멀리하는 프로그래밍 패러다임의 하나이다.

문이 아닌 식이나 선언으로 수행되는 선언형 프로그래밍 패러다임을 따르고 있다.

명령형이 아닌 선언형 프로그래밍 방식을 따르고 있다.

순수함수들은 오직 출력(return)만 수행하기 때문에, 입력된 값들이 변경되지 않는다.

따라서 최종 Output이 발생할 수 있도록 순수 함수들을 엮어서 호출해야 한다.

참고로,

명령형은 어떻게 풀어내는지(how to solve)에 집중하고,

선언형은 무엇을 풀어내는지(what to solve)에 집중한다.

**예를 들어, 최종적인 목표가 '물을 마시기' 라고 해보자.**

**(명령형) 어떻게(how) 할 것인지를 설명**

- 자리에서 일어난다.

- 부엌으로 이동한다.

- 컵과 물을 찾는다.

- 컵에 물을 부어 마신다.

**(선언형) 무엇(what)을 할 것인지를 설명**

- 물을 마셔보자.

**1. 사이드 이펙트를 없애야 한다.**

- 부작용이라는 뜻의 위 이름처럼 함수 외부의 어떤것을 변경할 시 의도치 않은 동작을 하는것입니다.

리액트에서 나오는 불변성을 강조하는 원칙이 여기에 적용 됩니다.

2. 참조 투명성 Referential Transparency

- 값의 참조등을 외부에서 받지않고 매개변수로 받아와 처리하는 방식으로 해야한다.

출처 : <https://medium.com/humanscape-tech/%ED%95%A8%EC%88%98%ED%98%95-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D%EC%97%90-%EA%B4%80%ED%95%B4-7f6172599fc>

<https://simsimjae.tistory.com/396>

**-절차지향 프로그래밍**

물이 위에서 아래로 흐르는 것처럼 순차적인 처리가 중요시 되며 프로그램 전체가 유기적으로 연결되도록 만드는 프로그래밍 기법이다. 절차 지향 프로그래밍은 컴퓨터 작업 처리 방식과 유사하기 때문에 객체 지향 언어를 사용하는 것에 비해 더 빨리 처리되어 시간적으로 유리하다는 인식이 일반적이지만, 소프트웨어 언어 발달과 컴파일러의 발달로 하드웨어가 소프트웨어의 발달을 따라가지 못하는 상황으로 변하고 있다.

\***포트란, 베이직 , C 언어 등이 대표적인 절차 지향 프로그래밍 언어에 속한다**

Top-Down 방식으로 설계된다.

비교적 작은 규모의 작업을 수행하는 함수(function)를 생성한다.

인수(parameter)와 반환값(value)으로 명령을 전달하고 수행한다.

객체 지향 프로그래밍과의 가장 큰 차이점으로 데이터와 함수를 별개로 취급한다.

특정 기능을 수행하려면 그 일을 해주는 메소드를 직접 호출해야 한다. (vs 객체 지향 프로그래밍 에선 특정 기능을 수행하는 메소드를 가진 객체를 만들어서, 그 인스턴스를 이용해 메소드를 호출한다.)

**단점**

유지보수가 어렵다.

정해진 순서대로 입력을 해야되므로 순서가 바뀌면 결과값을 도출하기 어렵다.

프로그램을 분석하기 어렵다.

대형 프로젝트에는 부적합하다.

출처 : <http://www.incodom.kr/%EC%A0%88%EC%B0%A8_%EC%A7%80%ED%96%A5>

- 객체지향 프로그래밍

OOP 라고 불리우는 객체지향 프로그래밍은

Object Oriented Programming의 약자로 객체의 관점에서 프로그래밍 하는 것

1. Object와 Class

Object, 즉 객체는 대상을 나타냅니다.

구성요소 하나하나를 객체라고 할 수 있습니다.

Class는 객체들이 공통적으로 갖는 속성을 모아서 정의 내린 것 입니다. 코드 작성의 기본 단위이며 객체들을 찍어내는 템플릿이라고 이해했습니다.

Class는 속성(attribute)와 행위(Method)를 정의 합니다.

**OOP의 4가지 특성**

**1) 추상화(Abstraction)**

자료 추상화는 불필요한 정보는 숨기고 중요한 정보만을 표현해 프로그램을 간단히 만듭니다.

**2) 캡슐화(Encapsulation)**

데이터와 메소드를 클래스 하나로 묶어 메소드로 접근 할 수 있도록 합니다. 또한 객체 내부에서 필요로 하는 정보를 은닉을 하는 특징 또한 존재 합니다. 저는 캡슐 알약과 비슷하다는 생각이 들었습니다. 안에 필요한 약 성분들을 담아서 은폐시키기 때문이죠!

**3) 상속(Inheritance)**

이미 정의 되어있는 상위 클래스와 메소드를 비롯한 모든 속성을 하위 클래스가 물려받습니다.

예를 들어, 동물이라는 클래스의 속성을 강아지 클래스가, 또 고양이 클래스가 물려 받을 수 있습니다.

상속이 필요한 이유는 코드의 중복을 막을 수 있어 코드가 더 간단해 집니다. 그럼 유지보수가 더 수월하겠죠?

**4)다형성(polymorphism)**

다형성은 형태가 같은데 다른 기능을 하는 것을 의미합니다.

부모 클래스로 부터 상속을 받지만, 자식클래스에서 물려받은 속성을 재정의 할 수 있습니다. 이를 오버라이딩(같은 이름의 메소드가 여러 클래스에서 다른 기능을 하는 것)이라고 합니다.

예를 들어, 슈퍼카 클래스가 일반카 클래스의 속도속성을 상속 받았지만, 속도 속성을 재정의 하여 슈퍼카의 속도에 맞게 재정의 할 수 있습니다.

**객체지향과 절차지향의 차이점 #**

객체지향 프로그래밍과 절차지향 프로그래밍이 일부 대조적인 특징을 가지고 있기 때문에 많은 사람들이 객체지향의 반대개념은 절차지향이라고 오해하고 있다. 위에서 설명한 바와 같이 절차지향은 데이터를 중심으로 순차적 실행에 초점을 맞추고 코드를 작성한다. 그렇다고 객체지향은 순차적으로 실행되는 것이 아니라고할 수 없다. 객체지향 프로그래밍도 절차지향 프로그래밍과 같이 절차적으로 실행되나 객체간의 관계에 초점을 두고 코드를 작성한다.

출처 : <https://velog.io/@hkoo9329/OOPObject-Oriented-Programming-%EA%B0%9D%EC%B2%B4-%EC%A7%80%ED%96%A5-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D-%EC%9D%B4%EB%9E%80>

**2. 함수형 프로그래밍이 갖는 장점과 단점은 무엇인가.**

**장점**

높은 수준의 추상화를 제공한다

함수 단위의 코드 재사용이 수월하다

불변성을 지향하기 때문에 프로그램의 동작을 예측하기 쉬워진다

장점들 중에서 불변성에 대한 것은 함수형 프로그래밍 자체의 장점이라기보다는 순수 함수(Pure Functions)를 사용함으로써 자연스럽게 따라오는 장점에 가깝다.

**단점**

순수함수를 구현하기 위해 코드의 가독성이 좋지 않을 수 있다. 재귀적 코드 스타일은 무한루프에 빠질 수 있다. 순수함수를 쓰는건 쉬울 수 있지만, 그것들을 조합하는 것은 쉽지 않다

출처 : <https://evan-moon.github.io/2019/12/15/about-functional-thinking/>

**3. 순수함수란 무엇인가.**

1. 동일한 인풋과 아웃풋으로 이뤄져있다.

- 실행시점이 변경 되어도 동일한 반환값을 반환한다. 항상 동일한 반환값을 반환하기 때문에 다루기 쉬운 함수가 된다.

2. 순수함수는 외부변수를 접근하지 않는다.

- 외부변수를 그대로 사용하지 않으며, 매게변수로 받아서 처리하는 방식으로 사용한다.

이러한 순수함수들을 조합해서 사용하는것이 함수형 프로그래밍 입니다.

출처 <https://velog.io/@kimu2370/redux%EC%9D%98-reducer%EA%B0%80-%EC%88%9C%EC%88%98%ED%95%A8%EC%88%98%EC%9D%B8-%EC%9D%B4%EC%9C%A0>