**1. Continuation**

- 우선 continuation을 이해하기 위해서 또, 이와 관련된 내용들을 이해하기 위해서는 continuation의 형태가 무엇인지 알아야 한다.

Continuation은 하나의 데이터 구조이다.(from Wikipedia) 즉, 어떠한 데이터를 담을 수 있는 하나의 틀인 것이다.

그렇다면, continuation은 무엇을 저장하는 데이터 구조일까? Youtube 강의에 의하면, continuation은 프로그램의 evaluation을 마치기 위해, 앞으로 해야 할 남은 일들을 말한다. (continuation just means the rest of the work that has to happen to finish evaluation of your program) 즉, continuation은 지금까지 수행한 결과들을 가지고 **앞으로 해야하는 계산 과정(computation)을 저장하는 데이터 구조**라고 할 수 있다.

Example)

전체 expression이 (- (+ (\* 3 4) 5) 8) 일 때, 아래 식들의 continuation(= k)을 구하라.

1. (- (+ (\* 3 4) 5) 8) K: (lambda (v) v)

2. (\* 3 4) K: (lambda (v) (- (+ v 5) 8))

3. 4 K: (lambda (v) (- (+ (\* 3 v) 5) 8))

Reference: Lecture slide(L 21), Youtube lecture

**2. call/cc**

- call/cc는 call with current-continuation의 약자이다. 직역하자면, 현재 continuation을 불러온다는 의미다. call/cc는 이를 호출하는 함수인 것이다. 이 call/cc는 함수를 인자로 넘겨 주어야 하는데, argument로 넘겨진 함수에는 현재 continuation이 담겨지게 된다.

Example)

(define retry #f)

(define factorial

(lambda (x)

(if (= x 0)

(call/cc (lambda (k) (set! retry k) 1))

(\* x (factorial (- x 1))))))

(factorial 3) ; Result -> 6

(retry 1) ; Result -> 6

(retry 2) ; Result -> 12

Reference: Lecture slide(L 24), Wikipedia

**3. Continuation-Passing Style**

- CPS는 함수가 값을 반환하지 않는 프로그래밍 스타일이다. 대신 CPS는 continuation으로 제어권을 넘긴다. CPS를 사용하면, 일반적인 recursion 함수와는 다르게 stack이 아니라, 앞으로 수행해야 할 내용이 담겨있어, 메모리 사용이 더 효율적이다. 또한, 프로그래머가 예외와 백트래킹, 스레드, 제네레이터(generator)등의 제어 구조를 추가할 수 있다

Example) \*ret은 현재 continuation으로 명명하였다.

function fact(n,ret) {

if(n == 0)

ret(1) ;

else

fact(n-1, function(t0) { ret(n \* t0) }) ;

}

fact (5, function(n) {

console.log(n); //콘솔에 결과 값(=120)이 출력된다.

})