배우는 내용

1. 프로그래밍언어 개요
2. 구문 구조-언어가 제공하는 의미와 형식 — 개요
3. 변수-속성, 바인딩, 타입검사..
4. 타입-문자열, 배열, 포인터, 구조적 타입도
5. 제어구조-수식, assignment, 반복문
6. 부프로그램, argument — 구조적 언어
7. 객체, 추상데이터타입
8. 함수형 (functional) 언어
9. 논리적 (logical)언어 — 새로운 개념

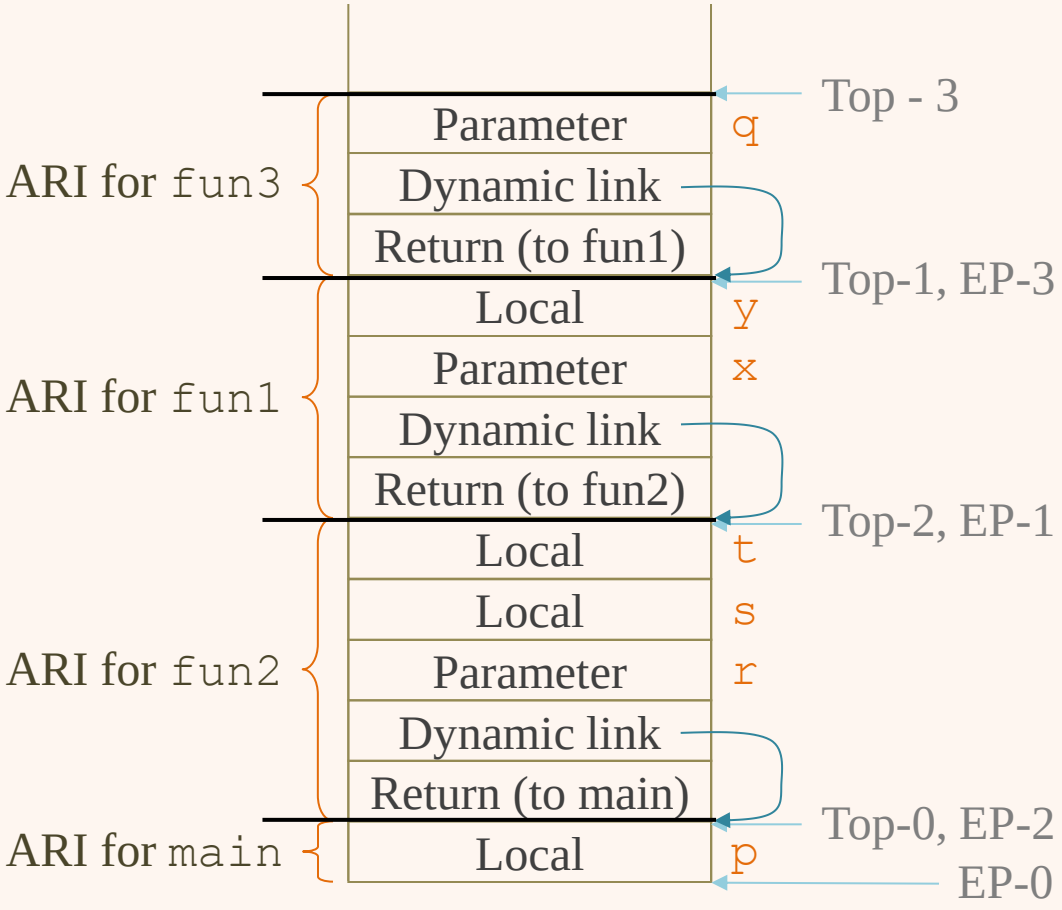
10 Subprogram 의 구현

- 호출하고 return 할 때의 작업
- 호출 :
  - 호출자(caller)의 실행상태 저장 (기억공간!)
  - actual parameter를 formal parameter로 전달 (기억공간!)
  - return할 주소를 피호출자(callee) 에게 전달 (기억공간!)
  - 제어를 피호출자(callee)에게 넘김
- return:
  - parameter가 out mode나 inout mode이면, 그 값을 actual parameter로 전달
  - subprogram이 함수이면 함수값을 caller에게 전달 (기억공간!)
  - 저장해둔 호출자(caller)의 실행 상태를 복원
  - 제어를 호출자(caller)에게 넘김

- 이 때 필요한 기억공간을 AR (activation record)라고 함
  - formal parameter, 지역변수, return address

Recursive Call이 없는 경우의 RTStack

```
void fun1(int x) {
  int y;
  ...          <-- 1
  fun3(y);
  ...
}

void fun2(float r) {
  int s, t;
  ...          <-- 2
  fun1(s);
  ...
}

void fun3(int q) {
  ...          <-- 3
}

void main() {
  float p;
  ...          <-- 0
  fun2(p);
  ...
}
```



dynamic link!!

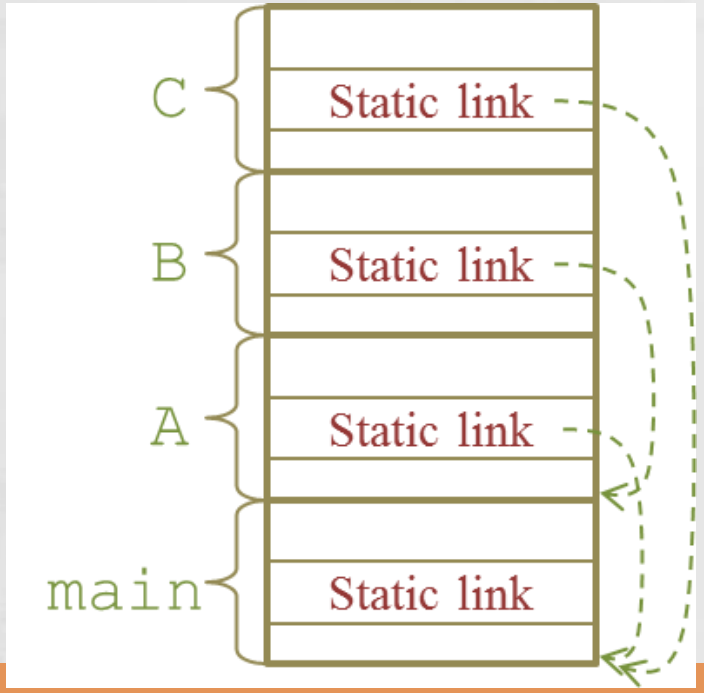
Nested subprogram에서는?

- 비지역변수 참조
  - scoping rule: SSR-2가지, DSR-2가지 ➔ 2x2 = 4가지

- SSR (Static Scoping Rule)

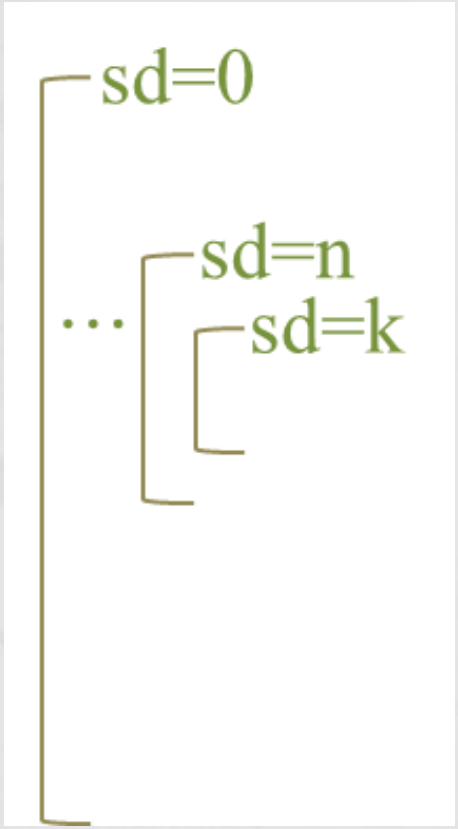
  (1) Static link 연결하고 따라가기

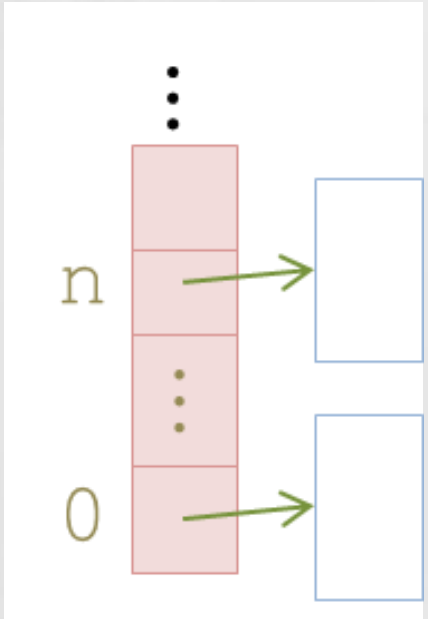
```
main ----  static-depth: 0
  A ------  static-depth: 1
    B -----  static-depth: 2
      call C
    end B
    call B
  end A
  C ------  static-depth: 1

  end C
  call A
end main
```



- SSR (Static Scoping Rule)
  - (2) Display

    각 depth 마다 AR 하나씩





- DSR (Dynamic Scoping Rule)

  (1) Deep access:

  Dynamic link (call chain) 그냥 따라가기

- DSR (Dynamic Scoping Rule)

  (2) Shallow access

```
void sub3() {
   int x, z;   x = u + v;   .
}

void sub2() {
   int w, x; ... sub3(); ..
}

void sub1() {
   int v, w; ... sub2(); ..
}

void main() {
   int u, v; ... sub1()
}
```

	sub1	sub3		sub2
main	main	sub2	sub3	sub1
u	v	x	z	w