# Q1. ABAP에서 DATA: 와 DATA : 의 차이가 있나요?

이 둘의 차이는 없음

# Q2. ABAP에서 "CALL FUNCTION" 과 "PERFORM"의 차이

CALL FUNCTION은 ***External Function을 호출하기 위해 사용한다.***
External Function은 독립적으로 존재하는 함수 모듈 OR 함수 그룹 내에 있는 함수 일 수 있다.

CALL FUNCTION 문은 호출된 함수를 실행하고 결과를 반환 받을 수 있다.
호출된 함수는 별도의 프로그램 유닛으로 간주되어, 독립적으로 실행된다.

PERFORM은 내부 서브루틴(Internal Subroutine)을 호출하기 위해서 사용한다.
내부 서브루틴은 호출하는 프로그램 내에서 정의된 서브루틴이다.
PERFORM문은 호출된 서브루틴을 실행하고 실행이 완료된 이후에 호출 지점으로 다시 돌아온다.
호출된 서브루틴은 호출한 프로그램의 실행 흐름을 공유하고 서브루틴 내에서 선언된 변수는 호출한 프로그램과 공유된다.

따라서 **CALL FUNCTION**은 외부 함수 호출에 사용되고 독립적으로 실행
**PERFORM**은 내부 서브루틴 호출에 사용되며, 호출한 프로그램과 실행 흐름을 공유함.

# Q3. ABAP에서 "FUNCTION" 과 "SUBROUTINE"의 차이

Function

- 함수는 독립적으로 실행 가능한 모듈화된 코드 조각
- 함수는 외부에서 호출되어 실행, 입력 매개 변수를 받고 결과를 반환
- 함수는 특정 작업을 수행하고 결과를 호출한 프로그램에 반환하는 역할
- 함수는 특정 기능을 수행하기 위해 사용되어지고, 일반적으로 입력 값을 받아 처리하고 결과를 반환

Subroutine

- 서브루틴은 호출한 프로그램 내에 정의된 코드 블록
- 서브루틴은 PERFORM문을 사용해서 호출
- 서브루틴은 호출한 프로그램과 동일한 실행 흐름을 공유
- 서브루틴은 호출한 프로그램의 변수를 공유, 로컬 변수를 가질 수도 있다.
- 서브루틴은 호출한 프로그램으로 돌아와서 계속 실행
- 서브루틴은 호출한 프로그램 내에서 재사용 가능한 코드 조각을 구성하는데 사용.

# Q4. DATA: 구문과 DATA구문의 차이

- DATA : 구문은 변수 선언과 초기화를 한 번에 처리하는 데 사용되는 키워드입니다.

- DATA 는 변수 선언만을 수행하는데 사용되는 키워드 입니다.
- DATA: 구문은 복수로 여러개의 변수들을 지정해줄 수 있습니다.
- DATA 구문은 단수로 하나의 변수만 지정해줄 수 있습니다.

예시

```
DATA : my_number TYPE i VALUE 10.

DATA my_number TYPE i.
my_number = 10.
```

# Q5. ABAP에서 APPEND 이해하기

- APPEND 구문은 내부 테이블에 새로운 레코드를 추가하는 작업을 의미

예시

```
DATA: lt_customers TYPE TABLE OF ty_customer,
      ls_customer TYPE ty_customer.

ls_customer-customer_id = 'C001'.
ls_customer-name = 'John Doe'.
ls_customer-age = 30.

APPEND ls_customer TO lt_customers.
```

위 예시에서 'lt_customers'는 내부테이블이고, 'ls_customer'는 내부 테이블에 추가할 새로운 레코드이다.
APPEND 명령문은 'ls_customer'의 데이터를 'lt_customers'에 추가해서 새로운 레코드를 생성한다.

# Q6. ABAP에서 SELECT SINGLE 구문이란?

- ABAP에서 'SELECT SINGLE'구문은 데이터 베이스 테이블에서 단일 레코드를 선택하는데 사용되는 구문
  이 구문을 사용하면 지정한 조건에 해당하는 첫 번째 레코드를 가져온다.
  다음은 'SELECT SINGLE' 구문의 일반적인 구조이다.

```
SELECT SINGLE <fields>
  FROM <table>
  [WHERE <condition>].
```

- 여기서 `<fields>`는 선택하려는 필드 목록을 나타내고, `<table>`은 데이터를 검색할 데이터베이스 테이블을 나타낸다. `<condition>`은 선택 조건을 지정하는 부분이며, 선택사항이다.

- `SELECT SINGLE` 구문을 사용할 때 주의해야 할 몇가지 사항

- SELECT SINGLE은 단일 레코드를 반환하므로 결과 집합이 비어있을 수 있다.
  따라서 조회 결과가 없을 경우 프로그램 실행 중에 오류를 처리할 수 있어야 한다.
- 조건을 지정하지 않으면 테이블을 첫 번째 레코드가 선택된다.
- SELECT SINGLE은 첫 번째 레코드만 선택하므로 인덱스를 활용해서 검색 성능을 향상 시킬 수 있다.

# Q7. ABAP에서의 TYPE 과 LIKE의 차이는 어떤것인가요?

TYPE

- TYPE의 경우 dictionary 에 있는 것을 프로그램영역에서 사용하고자 할 때 사용

LIKE

- LIKE의 경우 프로그램영역안에서 선언된 것을 사용하고자 할 때 사용

예를 들면
lt_intab type zt_testtable 은 zt_testtable 테이블을 프로그램영역의 internal table로 선언한것.
lt_intab_ln like line of lt_intab 은 프로그램영역안에서 선언된 lt_intab 과 동일한 타입을 가진 structure를 선언한 것.

굳이 이렇게 선언하지 않아도 되겠지만 (DB를 여러번 호출해야 하기 때문에)

이렇게 internal table로 올려놓고 사용을 하는 것.

# Q8. Assigning value(값 할당)이 뭔가요?

ABAP/4 에서는 변수를 선언하거나 실제 데이터가 처리되는 구문에서 Data Object에 값을 할당할 수 있다.
앞에서 살펴보았듯이 변수를 선언할 때 VALUE 구문을 통해서 초깃값을 정의할 수 있고, 셀제 데이터를 처리하는 프로그램 구문
내에서는 MOVE 또는 WRITE TO 구문을 이요앻서 변수에 값을 할당하게 된다.

```
<f2> = <f1>.
MOVE-CORRESPONDING <STRING1> TO <STRING2>.
WRITE <f1> TO <f2> [<option>].
```

MOVE TO 구문은 f2 = f1의 Equal(=) 구문과 동일한 의미이다.
Field String 값을 Copy 하는 경우에는 MOVE-CORRESPONDING 구문을 사용할 수 있다.
이때 string1과 string2의 필드명은 같지 않아도 된다.

MOVE-CORRESPONDING 구문은 구조체에서 값을 할당할 때 많이 사용되며, 각 구조체의 같은 필드에 값이 복사된다.
만약 구조체의 필드 이름 및 순서가 다른 경우 MOVE 구문을 사용하게 되면, 필드 이름에 관계없이 순서대로 할당이 일어나므로 주의해서 사용해야한다.

# Q9. LIKE 와 LIKE LINE OF의 차이점은 무엇인가요?

```
FIELD-SYMBOLS <fs> LIKE LINE OF gt_tab.
FIELD-SYMBOLS <fs> LIKE gt_tab.
```

이 두 가지 구문은 인터널 테이블에 HEADER LINE이 없을 때만 차이점이 있다.
(HEADER LINE이 있을 대는 두 구문의 기능은 같다.)
HEADER LINE이 없으면 LIKE LINE OF는 구조체의 Work Area를 선언하지만, LIKE 구문을 사용하게 되면 인터널 테이블을 선언하게 된다.

그러므로 HEADER LINE이 없는 인터널 테이블을 사용할 때는 LIKE 구문의 사용은 자제하는 것이 좋다.

# Q10. 프로그램 옆의 Description을 바꾸고 싶을 때 어떻게 하나요?

프로그램 옆의 Description을 변경하려면 T-CODE : SE80에서 좌측의 Object Tree에서 바꾸고자하는 해당 프로그램을 더블클릭 한 후에
상단의 메뉴중 이동 -> 속성을 선택하면 팝업창이 뜨는데 거기서 수정하면 된다.

Include 프로그램도 마찬가지로 똑같이 수정할 수 있다.