※ 처리 조건 (Implemenation requirements)

- Dynamic Scope Language
- non-local 변수의 참조는 `Deep Access` 방식
- 프로그램은 `구문 분석` 및 `실행` 의 단계로, 2단계로 동작
- 예약어(Preserved word)
  - `variable` : 함수의 지역변수를 정의
  - `call` : 다른 함수 호출 명령어
  - `print_ari` : 현재의 런타임 스택의 모든 내용을 화면에 출력하는 명령어
- 입력 파일에서 ASCII 코드 32이하인 모든 문자는 white-space로
  간주되며, white-space는 각 token을 구별하는 용도 이외에는 모두 무시된다.

세부 요건

- `구문 분석` 결과, 문법에 맞지 않으면, “Syntax Error.”라고 출력한 후 종료. 문법에 맞으면,
  “Syntax O.K.”를 출력하고, 프로그램에 명시된 기능을 수행.
- 모든 변수명과 함수명의 길이에는 제한이 없다.
- main이라는 이름을 갖는 함수는 반드시 한 개 존재
- 어떤 함수 내에서, n번째 실행문의 주소는 (function_name: n - 1)로 표시된다. 단, 지역변수 선언문은 실행문에 포함되지 않는다.
- `복귀 주소(Return Address)` 는 call 명령 다음의 주소를 가리킨다.
- 변수 참조시, 해당 변수의 link_count와 local_offset 값을 출력한다. 여기서 link_count는 동적 링크를 따라가는 (Traverse) 횟수를 의미

※ 오류 조건

- `실행` 단계
상황	처리	상세 설명
정의되지 않은 함수명 `call`	오류 메시지 출력 및 종료	(“Call to undefined the function name: function_name”)을 출력, 종료한다 (단, function_name은 정의되지 않은 함수명임).
정의되지 않은 변수명 호출	오류 메시지 출력 및 종료	(“Call to undefined the identifier: identifier”)을 출력, 종료한다 (단, identifier은 정의되지 않은 함수명임).
- `구문 분석` 단계
상황	처리	
main 함수가 정의 X	오류 메시지 출력 및 종료	(“No starting function.”)을 출력하고 종료
하나의 이름이 변수명과 함수명으로 동시에 사용	오류 메시지 출력 및 종료	(“Duplicate declaration of the identifier or the function name: ide (단, identifier/function_name은 중복 정의된 변수명 또는 함수명임)
같은 이름을 갖는 함수 2개 이상 정의	오류 메시지 출력 및 종료	(“Duplicate declaration of the function name: function_name”)을 (단, function_name은 중복 정의된 함수명임)
같은 이름을 갖는 지역 변수가 2개 이상 정의	오류 메시지 출력 및 오류 복구	(“Duplicate declaration of the identifier: identifier_name”)을 출력

개념 상 당연한 요구 조건

- 올바른 호출인 경우, 호출된 함수의 ARI (Activation Record Instance, Chapter 10)를 런타임-스택에 추가
- AR (Activation Record)의 구조는 다음과 같이 복귀주소 (Return Address), 동적 링크 (Dynamic Link), 그리고 지역변수 (Local Variables)들로 구성된다.
  - ![2022122620711.png](assets/스크린샷 2022-12-26 오전 2.07.11.png)
- ARI 내 각 항목의 크기는 1 word이며, 복귀주소와 동적 링크의 local_offset은 각각 0, 1이다. 또한, 지역변수의 local_offset은 2부터 시작한다

생각 해야할 것들

- 다음과 같이 문법 번형(EBNF)
- `<start> -> <functions>`
- `<functions> -> <function> { <functions> }`
- `<function> -> <identifier> { <var_definition> { <var_definition> } { <statement> } }`
- `<var_definition> -> variable <identifier> { , <identifier> } ;`

- `<statement> -> ( call` <identifier> | `print_ari` | <identifier>) ;
- `<identifier> ->` any names conforming to the C identifier standard