3.3.3

과거 yacc 컴파일러를 만드는 컴파일러. 요즘에는 이런 거 많음

3.4 생략

4.1

스크립트 언어들은 인터프리터가 바로 적용됨

많은 스크립트 언어들은 퓨어 인터프리터 대신 속도를 빠르게 하기 위해서 인터미디어 코드를 만든 다음 실행함

하이브리드 시스템이 실행속도가 아직도 늦기 때문에 실행될 때 컴파일 하는 방법을 씀. 자바도 마이크로 소프트도. -> 딜레이드 컴파일 시스템 (나중에 하기 때문에 다른 머신에서도 가능)

4.2

프로그래밍 언어는 대부분 패턴 매칭

예) index = total + sum; 키워드, 변수명, 함수명 등 전부 어휘

lexical analyzer는 syntax analyzer의 앞부분이다

렉시컬이 어휘를 찾으면 신텍스로 넘김

result = oldsum - value / 100;

8개의 어휘를 찾고 토큰을 부여 그리고 신텍스에 넘김

어휘가 아닌 것들은 전부 스킵한다. 예) 주석, 빈 줄

사용자가 만든 변수명이나 함수명 등은 symbol table에 등록이 됨

렉시컬 어날리시스를 만드는 방법 중 2번을 많이 씀

2. state transition diagram

state transition diagram은 방향성이 있는 그래프로, 기본적으로 state와 transition으로 구성

1. finit set of states 원으로 표시

2. set of transition 화살표로 표시

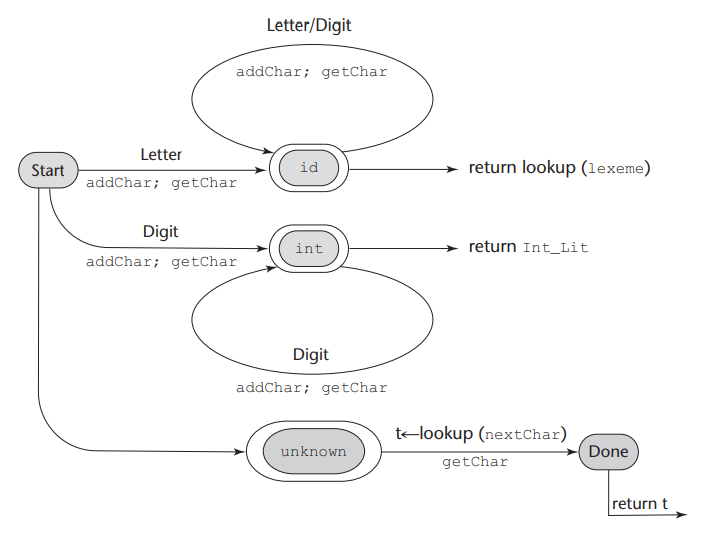
state중에 start state와 final state가 존재

주차장 자동문

차 확인 시간 오버

모터 작동 모터 작동

렉시컬 어날리시즈 에서는 input charartor가 액션임. 하나의 렉심을 찾으면 끝



4.3

parsing = syntax analyzing 로 하겠다.

parsing은 parse 트리를 그린다

syntax analysis의 두가지 목적이 있는데 하나는 문법적으로 올바르게 되는지 확인이다. 만약 error 가 발생하면 진단메시지를 만든다. 하지만 문제가 있다고 치고 뒤쪽의 error를 찾기 위해서 다음 것을 컴파일 한다.