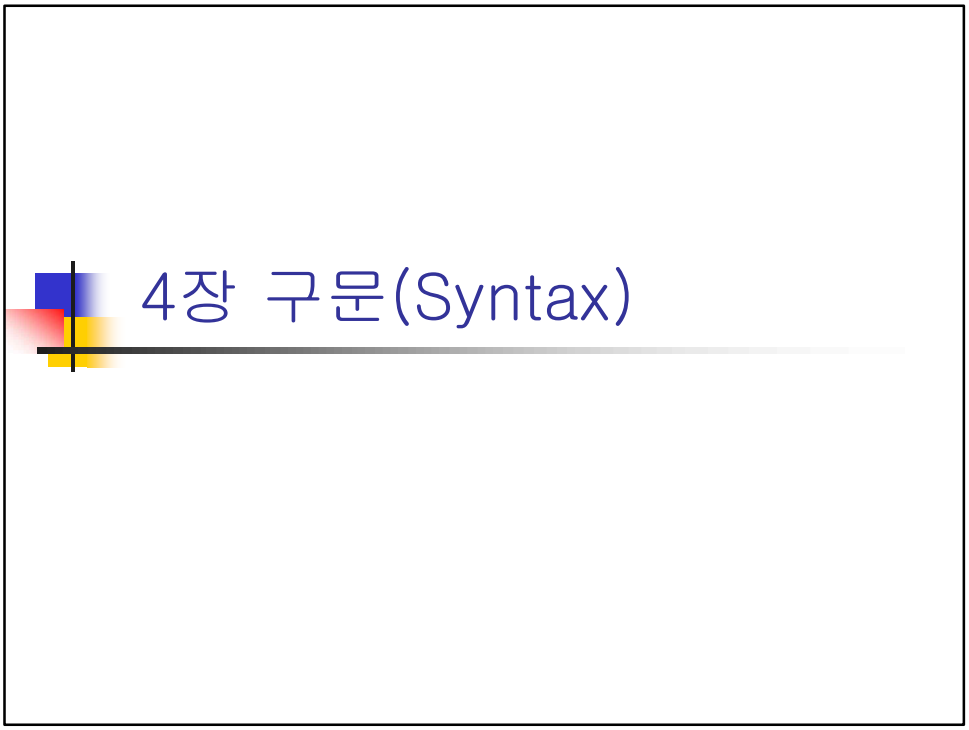
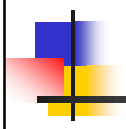

4장 구문(Syntax)

(프로그래밍) 언어의 구성

- 어휘구조(Lexical structure)
  - 단어의 구조
- 구문(Syntax)
  - 구문 구조
  - 문법을 이용해서 기술
    - Context-free grammar in BNF(Backus-Naur Form)
- 의미(Semantics)
  - 프로그램의 의미
  - 자연어를 이용한 기술
  - 수학적 기술
    - operational semantics, denotational semantics
    - axiomatic semantics



4.1 어휘구조(Lexical Structure)

어휘구조

- 어휘구조(Lexical structure)
  - 프로그래밍 언어를 구성하는 단어의 구조
- 토큰(Token) ?
  - 프로그래밍 언어를 구성하는 단어의 이름
- 토큰의 종류
  - 식별자(identifier)
    - X24, balance, putchar
  - 정수(integer)
    - 12, 350
  - 키워드(keyword)
    - if, while,
  - …

토큰 설계

- 언어의 토큰 정의
  - 언어의 의미 있는 모든 단어를 정의한다.
  - 토큰을 어떻게 표현할 수 있을까 ?
- 토큰 / 패턴 / 예

토큰	패턴	예
Keyword	**if, for, while, else, …**	
Symbol	<, <=, =, <>, >, >=	
Identifier	문자로 시작되는 문자 혹은 숫자들의 스트링	**x24, balance, putchar**
Integer	숫자들의 스트링	**314**
Literal	문자 스트링	**“test string”**

예제

- 예

```
if (i == j)
z = 0;
else z = 1;
```

- 이 문장을 구성하는 토큰들
  - Identifier : i,j,z
  - Keyword : if, else
  - Relation op: ==
  - Integer : 0, 1
  - Symbol : (,), =, ;
- 주의: (,),=,; 들도 토큰이다.

어휘 분석(Lexical Analysis)

- 무엇을 하고자 하는가 ?

```
if (i == j)
z = 0;
else z = 1;
```

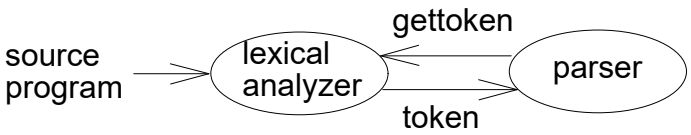
- 입력은 다음 문자 스트링이다.

```
₩tif (i == j)₩n₩t₩tz = 0;₩n₩telse₩n₩t₩tz = 1;
```

- 문제
  - 입력 스트링을 토큰들로 분리해야 한다.

토큰의 역할

- 어휘분석기(lexical analyzer)
  - 입력 스트링을 토큰 구성 규칙에 따라 토큰들로 분류한다.
- 어휘분석의 출력
  - 토큰들의 스트림
  - 파서(구문분석기)의 입력이 된다.



어휘분석 : 구현

- 어휘분석 구현은 다음을 해야 한다.
  - 1. 토큰에 해당하는 부스트링(substring)을 인식한다.
  - 2. 그 토큰의 부스트링을 "값" 혹은 "lexeme"으로 반환한다.
- 어휘분석기는 "관심 없는" 토큰들을 무시한다.
  - 공백, 주석 등