# 컴파일러 제작

컴파일러 제작의 이해를 돕기 위해 작성한 문서.

## 참고 자료

<http://www.kwangsiklee.com/2018/06/%ec%bb%b4%ed%8c%8c%ec%9d%bc%eb%9f%ac-%ea%b0%95%ec%9d%98-1-%ea%b0%9c%ec%9a%94/>

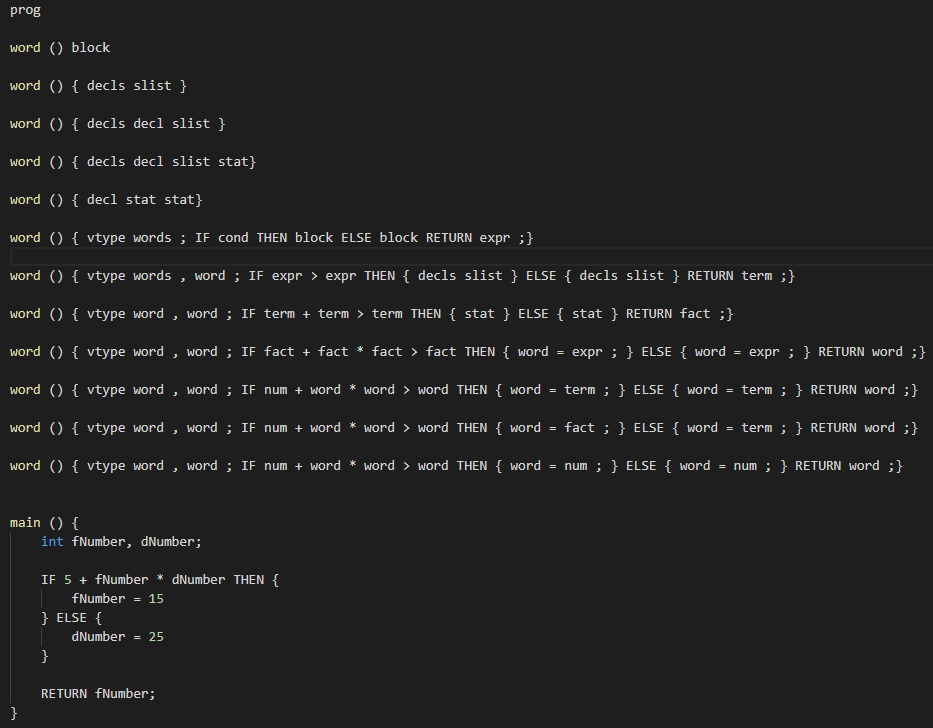
<https://crystalcube.co.kr/107?category=461606>

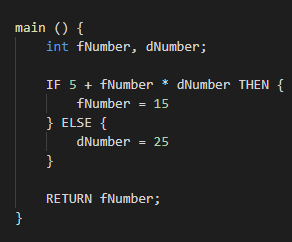
# PDF 이해

PDF의 표 1 GRAMMAR은 BNF 표기법에 따라 작성된 문법

왼쪽의 prog, decls, … 등은 토큰(token)이다.

이 외에 “{“, “(“, “;” 등에도 토큰 이름을 부여하여야 한다.



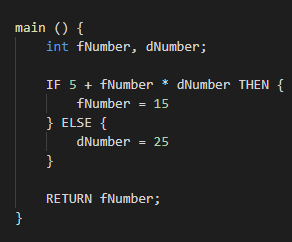


이런 코드가 있다고 가정하자.

# 단계

* Source Program
* Lexical Analyzer
* Syntax Analyzer
* Semantic Analyzer
* Intermediate Representation
* Code Optimizer
* Code generator
* Target Program

## Source Program



## Lexical Analysis

* 전처리
* 토큰 인식
* 기호표 구성
* 에러 처리에 대한 진단

### 전처리

스페이스(띄어쓰기), 줄바꿈 등을 모두 제거한다. (우리가 사용하는 문법은 스페이스에 대한 의미가 없다.)

다음은 전처리가 끝난 코드의 모습



### 토큰 인식



정규표현식을 사용하여 토큰 추출을 한다. 다음은 토큰 추출의 결과이다

word : main

구분자 : (

구분자 : )

block 구분자 : {

vtype : int

word : fNumber

구분자 : ,

word : dNumber

구분자 : ;

예약어 : IF

num : 5

연산자 : +

…

구분자 : ;

블록 구분자 : }

이런 형태가 된다.

### 기호표 구성

토큰끼리 의미 구분을 위해 토큰 번호와 토큰 값의 순서쌍으로 표현된 표이다.

(아직 수업에서 언급한 적이 없어서 보류)

## Syntax Analysis

### 문법

prog -> word () block

decls -> decls’

decls’ -> decl decls’ |

decl -> vtype words ;

words -> word words’

words’ -> , word words’ |

vtype -> int | char |

block -> { decls slist } |

slist -> slist’

slist’ -> stat slist’ |

stat -> IF cond THEN block ELSE block | WHILE cond block | word = expr ; | RETURN expr ;

cond -> expr expr’

expr’ -> ==expr | >expr

expr -> term term’

term’ -> + term |

term -> fact fact’

fact’ -> \*fact |

fact -> num | word

### FIRST (수정 필요)

First(prog) = First(word) = (a-z) | (A-Z)

First(decls) = First(decls’) = First(decl) = First(vtype) = int | char | ε

First(words) = First(word) = (a-z) | (A-Z)

First(words’) = ,

First(block) = { | ε

First(slist) = First(slist’) = First(stat) = First(word) ∪ (IF | WHILE | RETURN | ε)

First(cond) = First(expr) = First(term) = First(fact) = First(num) ∪ First(word) = (a-z) | (A-Z) | (0-9)

### FOLLOW (수정 필요)

Follow(prog) = $

Follow(decls’) = Follow(decls) = First(slist) ∪ Follow(slist) =

Follow(decl) = int | char | ε

Follow(words) = Follow(words’) = ;

Follow(vtype) = First(words) = (a-z) | (A-Z)

Follow(block) = Follow(prog) ∪ Follow(stat) ∪ ELSE = $ | IF | WHILE | RETURN | ε | (a-z) | (A-Z) | } | ELSE

Follow(slist) = Follow(slist’) = }

Follow(stat) = First(stat) ∪ Follow(slist) = IF | WHILE | RETURN | ε | (a-z) | (A-Z) | }

Follow(cond) = THEN ∪ First(block) = THEN | {

Follow(expr) = ( > | ==) ∪ Follow(cond) ∪ ; = ( > | == | THEN | { | ; )

Follow(term) = Follow(expr) ∪ + = ( > | == | THEN | { | ; | + )

Follow(fact) = Follow(term) ∪ \* = ( > | == | THEN | { | ; | + | \* )

Follow(word) = Follow(fact) ∪ (( | ; | , | =) = ( > | == | THEN | { | ; | + | \* | ( | ; | , | = )

Follow(num) = Follo(fact) = ( > | == | THEN | { | ; | + | \* )