어휘분석

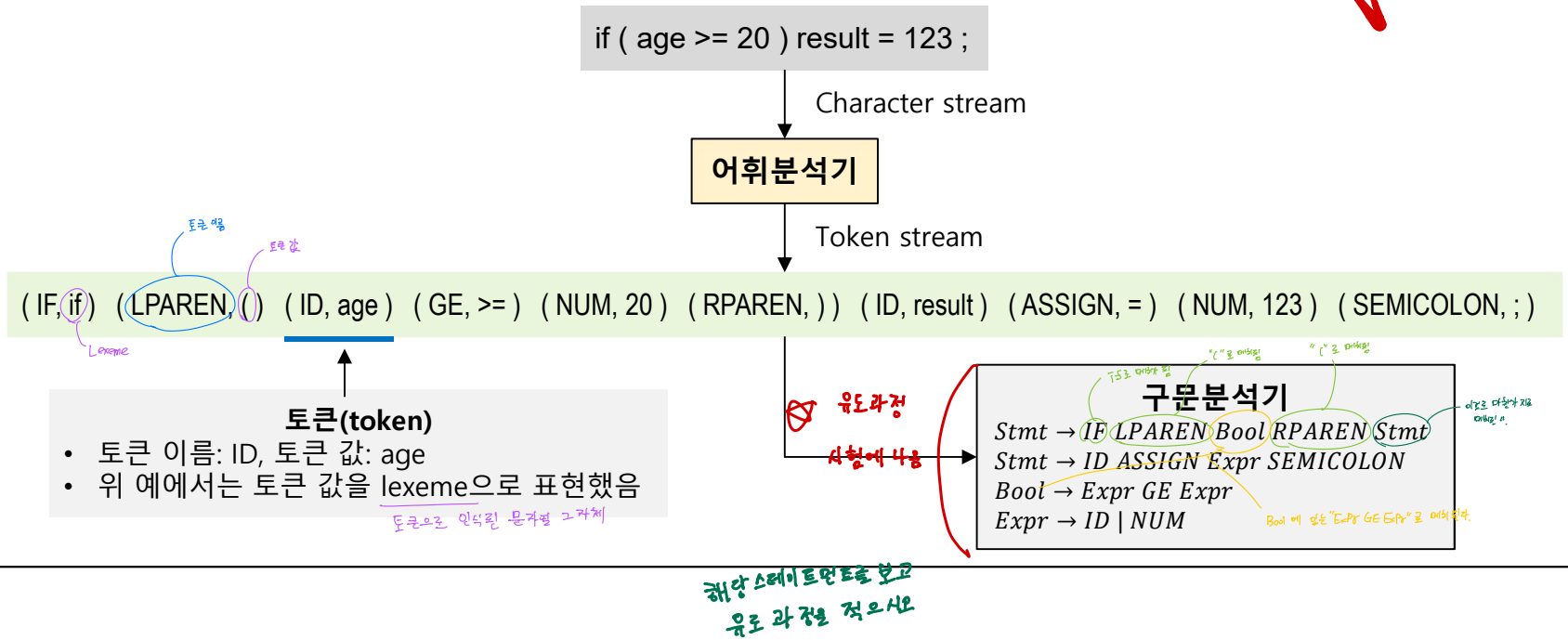
목차

- 어휘 분석기 개요
- 토큰
  - 렉심
  - 패턴
- 어휘 분석과 심볼테이블
- 어휘 분석 연습
- 키워드, 연산자, 식별자, 리터럴, 구분자

어휘분석기 (1/2)

어휘분석기(lexical analyzer, tokenizer, scanner, lexer)

- 어휘분석기는 소스코드를 문자들의 나열로 입력 받아 토큰들의 나열을 출력하며 일반적으로 주석(comment)과 whitespace 문자들(space, \t, \n 등)은 토큰으로 처리하지 않고 무시
- 토큰(**token**)은 토큰 이름(**token name**)과 토큰 값(**token attribute value**, token value)의 두 요소로 구성
- 렉심(**lexeme**)은 하나의 토큰에 대응하는 문자열



어휘분석기 (2/2)

어휘분석기(lexical analyzer, tokenizer, scanner, lexer)

- 어휘분석기는 소스코드를 문자들의 나열로 입력 받아 토큰들의 나열을 출력. 일반적으로 주석(comment)과 whitespace 문자들(space, \t, \n 등)은 토큰으로 처리하지 않고 무시됨
- 렉심(**lexeme**)은 하나의 토큰을 구성하는 문자열
- 토큰(**token**)은 토큰 이름(**token name**)과 토큰 값(**token attribute value**, token value)의 두 요소로 구성
- 토큰 이름은 파서(parser)가 수행하는 구문분석에 사용되며, 프로그래밍언어의 문법에서 터미널 심볼(terminal symbol)에 해당
- 토큰 값은 일반적으로 토큰에 대한 정보들(lexeme, type, line number 등)이 저장된 심볼테이블(**symbol table**) 내 엔트리(entry)로의 포인터
- 대부분 프로그래밍언어에서의 토큰들은 키워드(**keyword**), 연산자(**operator**), 식별자(**identifier**), 리터럴(**literal**), 구분자(**delimiter**)의 5개 부류로 나뉨
- 패턴(**pattern**)은 토큰에 해당하는 lexeme(들)의 문자 구성 방식을 기술한 것이며 정규표현식(**regular expression**)으로 기술됨

토큰 이름	Lexeme	Pattern
IF	if	if
LPAREN	(	(
GE	>=	>=
ID	age, result	[a-zA-Z][a-zA-Z0-9]*
NUM	20, 123	[0-9]+

`if ( age >= 20 ) result = 123 ;`

Character stream

어휘분석기

Token stream

`( IF, if ) ( LPAREN, ( ) ( ID, age ) ( GE, >= ) ( NUM, 20 ) ( RPAREN, ) ) ( ID, result ) ( ASSIGN, = ) ( NUM, 123 ) ( SEMICOLON, ; )`

토큰 부류	예
키워드	if, for, while
연산자	+, *, >=
식별자	age, score1
리터럴	123, "Korea"
구분자	(,), {, }, ;

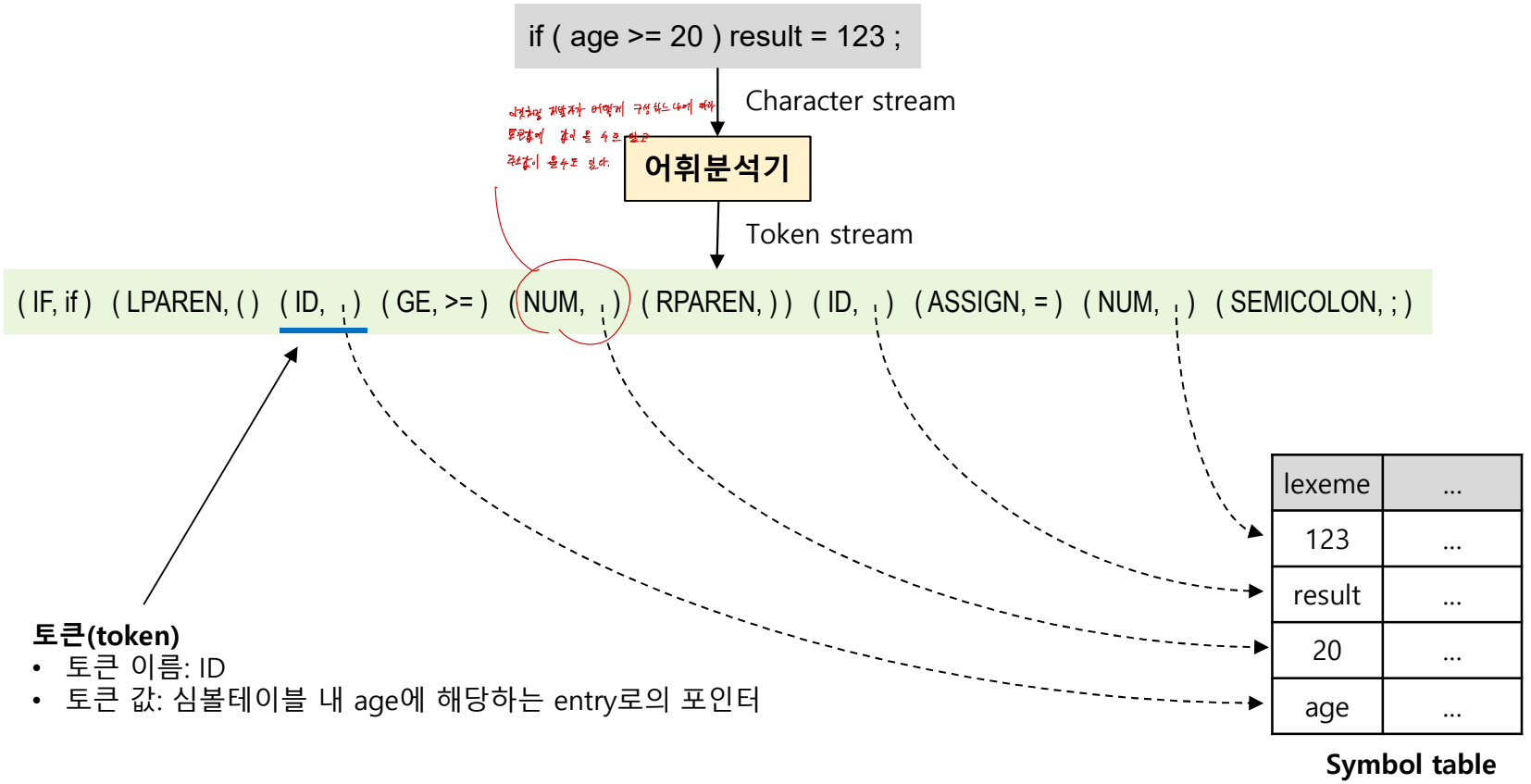
토큰(token)

- 토큰 이름: ID, 토큰 값: age
- 위 예에서는 토큰 값을 lexeme으로 표현했음

구문분석기

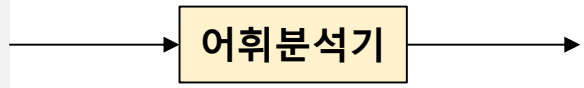
$Stmt \rightarrow IF\ LPAREN\ Bool\ RPAREN\ Stmt$
$Stmt \rightarrow ID\ ASSIGN\ Expr\ SEMICOLON$
$Bool \rightarrow Expr\ GE\ Expr$
$Expr \rightarrow ID \mid NUM$

어휘 분석과 심볼테이블


if (age >= 20) result = 123 ;
Character stream
어휘분석기
Token stream
(IF, if) (LPAREN, ( ) (ID,) (GE, >=) (NUM,) (RPAREN,)) (ID,) (ASSIGN, =) (NUM,) (SEMICOLON, ;)
토큰(token)
• 토큰 이름: ID
• 토큰 값: 심볼테이블 내 age에 해당하는 entry로의 포인터
lexeme
...
123
result
20
age
Symbol table

어휘 분석 연습 1

```
int fact(int n){
  if(n==0) return 1;
  return n*fact(n-1);
}
```



(INT, **int**)
(ID, **fact**)
(LPAREN, **(**)
(INT, **int**)
(ID, **n**)
(RPAREN, **)**)
(LBRACE, **{**)
(IF, **if**)
(LPAREN, **(**)
(ID, **n**)
(EQ, **==**)
(NUM, **0**)
(LPAREN, **)**)
(RETURN, **return**)
(NUM, **1**)
(SEMICOLON, **;**)
(RETURN, **return**)
(ID, **n**)
(MUL, *****)
(ID, **fact**)
(LPAREN, **(**)
(ID, **n**)
(SUB, **-**)
(NUM, **1**)
(RPAREN, **)**)
(SEMICOLON, **;**)
(RBRACE, **}**)

Lexeme

어휘 분석 연습 2

```
int scanf(const char *format, ...);
int printf(const char *format, ...);

int fact(int n){
   int result=1;
   for(int i=1; i<=n; i++) result*=i;
   return result;
}
int main(){
   int n;
   scanf("%d", &n);
   if(n<0) printf("wrong number₩n");
   else printf("fact(%d)=%d", n, fact(n));
   return 0;
}
```

→ **어휘분석기** →

```
...
( FOR, for )
...
( MUL_ASSIGN, *= )
...
( INT, int )
( ID, main )
( LPAREN, ( )
( RPAREN, ) )
( LBRACE, { )
( INT, int )
( ID, n )
( SEMICOLON, ; )
( ID, scanf )
( LPAREN, ( )
( STR, "%d" )
( COMMA, , )
( AMPERSAND, & )
( ID, n )
( RPAREN, ) )
( SEMICOLON, ; )
...
( ELSE, else )
( ID, printf )
( LPAREN, ( )
( STR, "fact(%d)=%d" )
( COMMA, , )
...
```

Lexeme

콤마는 한 문자 이다

ID임

어휘 분석 연습 3

```
int printf(const char *format, ...);

int main(){
   double score[]={1.2, 3.4};
   int value=score[0];
   int *p=&value;
   printf("%d", *p);
   return 0;
}
```

...
(DOUBLE, **double**)
(ID, **score**)
(LBRACKET, **[**)
(RBRACKET, **]**)
(ASSIGN, **=**)
(LBRACE, **{**)
(NUM_REAL, **1.2**)
(COMMA, **,**)
(NUM_REAL, **3.4**)
(RBRACE, **}**)
(SEMICOLON, **;**)
(INT, **int**)
(ID, **value**)
(ASSIGN, **=**)
(ID, **score**)
(LBRACKET, **[**)
(NUM_INT, **0**)
(RBRACKET, **]**)
(SEMICOLON, **;**)
(INT, **int**)
(ASTERISK, *****)
(ID, **p**)
(ASSIGN, **=**)
(AMPERSAND, **&**)
(ID, **value**)
(SEMICOLON, **;**)
...

Lexeme

키워드, 연산자, 식별자, 리터럴, 구분자

키워드(keyword), 예약어(reserved word)
- 예약어는 지정된 혹은 (현재는) 지정되지 않은 용도를 위해 예약된 이름으로, 식별자로 사용 불가
- 키워드는 특정 문맥에서 특정한 의미를 갖는 단어로, 일반적으로 키워드 집합은 예약어 집합의 부분집합
- Java 키워드: int, double, char, void, if, else, for, while, break, continue, return, class, extends, public, private, static, …

연산자(operator)
- 피연산자(operand)에 대해 적용될 특정 동작을 표현하는 심볼(Java 예 → - + * / % > >= < <= != == && ||)

식별자(identifier)
- 프로그래밍언어 내 개체(예: 변수, 함수, 레이블 등)를 명명하는 문자열
- 대부분, 숫자로 시작하지 않는 alphanumeric 문자들의 나열(예 → age, score, city, flag, setFlag)

리터럴(literal)
- 소스코드 내 값에 대한 표현으로, 대부분 PL에서, 정수, 실수, 문자, 문자열, 불리언의 특정 값이 소스코드에 표현된 것
- 예 → 23, 3.85, 'A', "Pusan"

구분자(delimiter)
- 코드 내 영역 구분을 위해 사용되는 심볼
- Java 예 → ( ) { } [ ] ; 등

Java Code

```
public class Test {
   public static void main(String[] args) {
      int age=23;
      double score=3.85;
      String city="Pusan";
      char flag=setFlag(age);
      System.out.println(age+","+score+","+city+","+flag);
   }
   private static char setFlag(int age) {
      if(age>=20) return 'A';
      else return 'B';
   }
}
```

References

- 박두순. (2016). (내공 있는 프로그래머로 길러주는)컴파일러의 이해. 한빛아카데미.
- 창병모. (2021). 프로그래밍 언어론 : 원리와 실제. 인피니티북스.
- Aho, A., Lam, M., Sethi, R., Ullman, J. (2006). Compilers: Principles, Techniques, and Tools (2nd Ed.). Addison Wesley.
- Nystrom, R. Crafting interpreter. https://craftinginterpreters.com/
- Sebesta, R. (2012). Concepts of Programming Languages (10th. ed.). Pearson.
- Thain, D. (2023). Introduction to Compilers and Language Design.
- Paxson, V. (1995). Flex, version 2.5.
- Donnelly, C., Stallman, R. (2008). Bison.
- LexAndYacc.pdf (epaperpress.com)
- Tom Niemann, LEX & YACC. http://epaperpress.com/lexandyacc