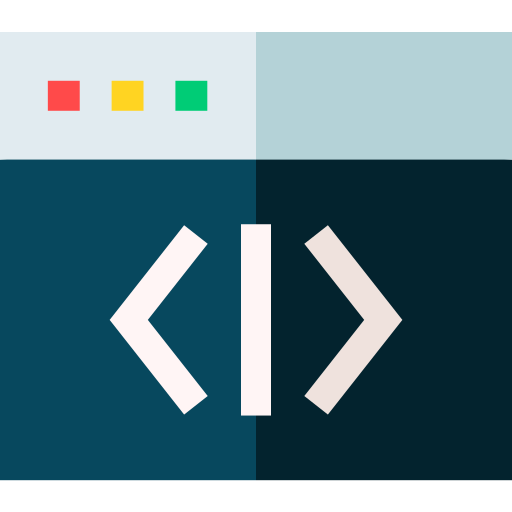
**프로그래밍 언어론**



**2020.05.03**

20151167 이인재

#1 3장 연습문제

7. 예제 3.4의 문법을 사용하여 다음 각 문장에 대해 파스 트리와 최좌단 유도를 보여라.

**예제 3.4 식에 대한 모호하지 않은 문법**

**<assign> -> <id> = <expr>**

**<id> -> A | B | C**

**<expr> -> <expr> +<term>**

**| <term>**

**<term> -> <term> \* <factor>**

**|<factor>**

**<factor> -> (<expr>)**

**|<id>**

b. A = B \* C +A

b-1) 파스 트리

<assign>

<id> = <expr>

A <expr> + <term>

<term> <factor>

<term> \* <factor> <id>

<factor> <id> A

<id> C

B

b-2) 최좌단 유도

<assign> -> <id> = <expr>

<id> -> A = <expr >

A = <expr> -> A = <expr> + <term>

A = <expr> + <term> -> A = <term> + <term>

A = <term> + <term> -> A = <term> \* <factor> + <term>

A = <term> \* <factor> + <term> -> A = <factor> \* <factor> + <term>

A = <factor> \* <factor> + <term> -> A = <id> \* <factor> + <term>

A = <id> \* <factor> + <term> -> A = B \* <factor> + <term>

A = B \* <factor> + <term> - > A = B \* <id> + <term>

A = B \* <id> + <term> - > A = B \* C + <term>

A = B \* C + <term> -> A = B \* C + <factor>

A = B \* C + <factor> -> A = B \* C + <id>

A = B \* C + <id> -> A = B \* C + A

c. A = A+ ( B \* C )

c-1)파스 트리

<assign>

<id> = <expr>

A <expr> + <term>

<term> <factor>

<factor> (<expr>)

<id> (<term>)

A (<term> \* <factor>)

<factor> <factor>

<id > <id>

B C

c-2)최좌단 유도

<assign> -> <id> = <expr>

<id> = <expr> -> A = <expr>

A = <expr> -> A = <expr> + <term>

A = <expr> + <term> -> A = <term> + <term>

A = <term> + <term> -> A = <id> + <term>

A = <id> + <term> -> A = A + <term>

A = A + <term> -> A = A + <factor>

A = A + <factor> -> A = A + (<expr>)

A = A + (<expr>) -> A = A + (<term>)

A = A + (<term>) -> A = A + (<term> \* <factor>)

A = A + (<term> \* <factor>) -> A = A + (<factor> \* <factor>)

A = A + (<factor> \* <factor>) -> A = A + (<id> \* <factor>)

A = A + (<id> \* <factor>) -> A = A + ( B \* <factor>)

A = A + ( B \* <factor>) -> A = A + ( B \* <id>)

A = A + ( B \* <id>) -> A = A + ( B \* C)

8. 다음 문법이 모호함을 증명하라

**<S> -> <A>**

**<A> -> <A> \* <A> | <id>**

**<id> -> x | y | z**

일반적으로 문법이 모호하다는 것은 파스 트리의 형식이 두개 이상으로 나뉘어 진다는 것으로 증명할 수 있다.

본 문법에서는 x\*y\*z라는 문장을 표현할 때, <A> \* <A> -> <A> \* <A> \* <A>로 표현하는 과정에서

(<A> \* <A>) \* <A> 와 <A> \* (<A> \* <A>)이 우선순위를 판단하는데 있어서 모호하다고 볼 수 있다.

<S>

<A>

<A> \* <A>

<A> \* <A> <id>

<id> <id> z

X y

<S>

<A>

<A> \* <A>

<id> <A> \* <A>

x <id> <id>

y z

9. 예제 3.4의 문법에서, 우선순위가 +나 \*보다 높은 단항 연산자 -를 추가하여 그 문법을 수정하라

**<assign> -> <id> = <expr>**

**<id> -> A | B | C**

**<expr> -> <expr> +<term>**

**| <term>**

**<term> -> <term> \* <term2>**

**|<term2>**

**<term2> -> <term2> - <factor>**

**|factor**

**<factor> -> (<expr>)**

**|<id>**

연산자 우선순위는 파스 트리의 가장 낮은 부분이 가장 높은 우선 순위를 갖고 있으므로 따라서 \*보다 낮은 부분에 -연산을 넣어주면 된다.

10. 다음 문법에 의해 정의된 언어를 우리말로 기술하라.

<S> -> <X><Y>

<X> -> x<X> | x

<Y> -> y<Y> | y

시작기호인 <S>는 <X>와 <Y>를, <X>와 <Y>는 각각 x<x>|x 와 y<Y>|y의 하위계층을 갖고 있으며 이는 x와 y가 반복을 수행한다는 의미를 갖고 있다.

따라서 S|S는 {xx+yy+}의 패턴을 갖는 문자열의 언어이다.

12. 다음 문법을 생각하자.

<S> -> a<S>c<B> | <A> | b

<A> -> c <A> | c

<B> -> d |<A>

다음 문장들 중에서 어느 것이 위의 문법에 의해 생성된 언어에 속하는가?

a. abbccd

<S> -> a<S>c<B>

a<S>c<B>

이 부분에서 <S>를 호출시 abbccd라는 문장은 나올 수가 없으므로 a는 생성될 수 없다.

b. acccbda

<S> -> a<S>c<B>

-> a<A>c<B> -> ac<A>c<B> -> accc<B> -> acccd

이 부분에서 acccbda라는 문장이 나올 수 없으므로 b는 생성될 수 없다.

c. acccbcbccc

<S> -> a<S>c<B>

-> a<A>c<B> -> ac<A>c<B> -> acc<A>c<B> -> accc<A>c<B> -> acccbc<B>

이 부분에서 acccbcbcc 라는 분장이 나올 수 없으므로 c는 생성될 수 없다.

d. acddaccd

<S> -> a<S>c<B>

-> a<A>c<B> -> ac<A>c<b> -> ac<A>c<b>

이 부분에서, acddaccd 라는 문장이 나올 수 없으므로 이는 생성될 수 없다.

e. acccdc

<S> -> a<S>c<B>

-> a<A>c<B> -> ac<A>c<B> -> accc<B>

이 부분에서 acccdc 라는 문장이 나올 수 없으므로 결과적으로 모든 문장이 속하지 않는다.

13. 문자 a가 n번 나오고, 다음에 문자 b가 n번 나오는 스트링들로 구성된 언어에 대한 문법을 작성하라.

여기서 n> 0이다. 예를 들면, 스트링 abb, aabbbb,aaaabbbbbbbb 등은 언어에 속하지만 a, aabb,ba,aaabb등은 속하지 않는다.

이는 자세히 살펴보면 b가 a의 두배만큼의 출력이 이루어 짐을 확인할 수 있다. 따라서

<S> -> a<S>bb | abb와 같이 표현할 수 있다.

abb = <S> -> abb

aabbbb = <S> -> a<S>bb -> aabbbb

aaaabbbbbbbb= <S> -> a<S>bb -> aa<S>bbbb ->aaa<S>bbbbbb -> aaaabbbbbbbb

이와 같이 출력이 가능함을 알 수 있다.

15. 예제 3.1의 BNF를 EBNF로 변환하라.

BNF와 EBNF의 차이점은 식을 표현함에 있어 간소화를 추구했다는 부분이다.

따라서 선택사항을 [ ], RHS의 선택사항을 ( ), 반복사항을 { }로 표현했음을 알 수 있다.

**BNF**

<program> -> begin <stmt\_list> end

<stmt\_list> -> <stmt>

| <stmt> ; <stmt\_list>

< stmt > -> <var> = <expression>

< var > -> A | B | C

<expression> -> <var> + <var

|.<var> - <var>

| <var>

**EBNF**

<program> -> begin <stmt\_list> end

<stmt\_list> -> <stmt>{ ; <stmt\_list> }

<stmt> -> <var> = <expression>

< var > -> ( A | B | C )

<expression> -> <var>{ (+|-) <var> }

**#2**

**다음 C 언의 구조에 대한 구문을 BNF와 EBNF로 각각 작성하라. C 언어의 설계 사항을 참조하라.**

* + While 문

**BNF 표현**

<assign> -> <while\_statement>

<while\_statement> -> while ( <expr> ) <statement>

<expr> -> <expr><expr> | <op> <var>; | <var> <op> <var> | <var> <op> <var> ; |

<var> -> <id> | <num>

<id> -> i | sum

<num> -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 ….. | 100

<op> -> + | - | \* | / | % | < | > | <= | >= | = | != | ++ | -- | ==

<statement> -> { <expr> }

**EBNF 표현**

<assign> -> <while\_statement>

<while\_statement> -> while ( <expr> ) <statement>

<expr> -> one of <op> <var>; <var> <op> <var> <var> <op> <var>; {,<expr>}

<var> -> one of <id> <num>

<id> -> one of i sum

<num> -> one of 0 1 2 3 4 5 6 7 8 9 … 100

<op> -> one of + - \* / % < > <= >= = != ++ -- ==

<statement> -> { <expr> }

**EBNF로 표현함에 있어 결과적으로 |라는 표시를 one of라는 문구로 통일 시켰으며 반복구조에 대해서는 { }를 이용하여 최소 한번 이상 표현하도록 선언하였다.**

**위에서 BNF로 작성한 구문 구조를 이용하여 다음 C 문장에 대한 최좌단 유도와 파스 트리를 각각 보여라.**

* + **while (i <100) {**

**sum = sum + i;**

**i = i + 1;**

**}**

**파스 트리**

<assign>

<while\_statement>

While( <expr> ) <statement>

<var> <op> <var> { <expr> }

<id> < <num> <expr> <expr>

i 100 <var><op><var> <expr> <expr>

<id> = <id> <op> <var>; <expr> <expr>

sum sum + <num> <var><op><var> <op> <var> ;

1 <id> = <id> + <num>

i i 1

**최좌단 유도**

<assign> -> <while\_statement>

<while\_statement>

-> while (<expr>) <statement>

-> while (<var> <op> <var>) <statement>

-> while <<id> <op> <var>> <statement>

-> while ( i <op> <var>) <statement>

-> while ( i = <var>) <statement>

-> while ( i = <num>) <statement>

-> while ( i = 100 ) <statement>

-> while ( i = 100 ) { <expr> }

-> while ( i = 100 ) { <expr><expr><expr><expr> }

-> while ( i = 100 ) { <var> <op> <var> <expr><expr><expr> }

-> while ( i = 100 ) { <id> <op> <var> <expr><expr><expr> }

-> while ( i = 100 ) { sum <op> <var> <expr><expr><expr> }

-> while ( i = 100 ) { sum = <var> <expr><expr><expr> }

-> while ( i = 100 ) { sum = sum <expr><expr><expr> }

-> while ( i = 100 ) { sum = sum <op><var>;<expr><expr> }

-> while ( i = 100 ) { sum = sum + <var>;<expr><expr> }

-> while ( i = 100 ) { sum = sum + <id>;<expr><expr> }

-> while ( i = 100 ) { sum = sum + i; <expr><expr> }

-> while ( i = 100 ) { sum = sum + i; <var><op><var><expr> }

-> while ( i = 100 ) { sum = sum + i; <id><op><var><expr> }

-> while ( i = 100 ) { sum = sum + i; i<op><var><expr> }

-> while ( i = 100 ) { sum = sum + i; i = <var><expr> }

-> while ( i = 100 ) { sum = sum + i; i = <num><expr> }

-> while ( i = 100 ) { sum = sum + i; i = i<expr> }

-> while ( i = 100 ) { sum = sum + i; i = i <op> <var>; }

-> while ( i = 100 ) { sum = sum + i; i = i + <var>; }

-> while ( i = 100 ) { sum = sum + i; i = i + <num>; }

-> while ( i = 100 ) { sum = sum + i; i = i + 1; }

End

이와 같이 최좌단 유도가 이루어짐을 확인할 수 있다.