5주차 1차시 프로그래밍 언어의 개념

【학습목표】

1. 프로그래밍 언어의 개념과 번역기의 종류, 그리고 역할을 살펴보고 설명할 수 있다.
2. 프로그래밍 언어의 역사와 구조에 대해 살펴보고 설명할 수 있다.

학습내용1 : 프로그래밍 언어의 번역기

소프트웨어는 프로그래밍 언어로 작성된 프로그램의 집단을 의미하는 것을 강조
과거의 프로그래밍 언어는 복잡하고 전문가만이 작성하였지만, 오늘날에는 비전문가도 쉽고 간단하게 작성할 수 있다는 점을 강조

1. 프로그래밍 언어의 개념

* 컴퓨터에 사용하기 위해서 만든 인공 언어를 프로그래밍 언어라 한다.

① 프로그래밍 언어의 계층
- 컴퓨터가 이해할 수 있는 유일한 언어는 0과 1로 구성된 기계어뿐

- 기계어와 자연어의 중간 형태로 개발된 인공 언어가 프로그래밍 언어 (저급언어+고급언어)
- 저급언어는 기계어와 어셈블리어로 구성 (기계어와 1:1로 대응시켜 기호화 한 언어)

- “15번지 내용을 17번지의 내용에 더하여 19번지에 저장하는 프로그램”을 기계 언어와 어셈블리 언어로 나타내 보자.

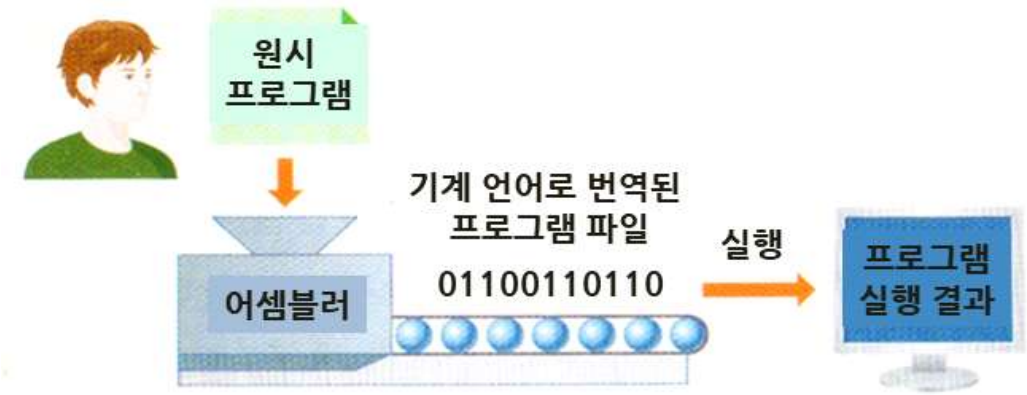
[기계 언어 프로그램]		[일대일 대응]	[어셈블리 언어 프로그램]	
<명령>	<실행 대상>		<명령>	<실행 대상>
0000 0010	0000 1111	↔	LOAD	15
0000 1000	0001 0001	↔	ADD	17
0000 0100	0001 0010	↔	STORE	19

기계 언어 명령의 실행 대상을 이진수에서 십진수로 변환해 보면 각각 15, 17, 19가 되어 어셈블리 언어의 실행 대상과 일대일로 대응된다.

② 저급언어(Low level Language)

- 기계어 중심 언어로 컴퓨터가 명령을 직접 인식하기 때문에 처리 속도는 빠르지만, 하드웨어에 의존적이다.
  - 기계어
    : 직접 실행하므로 처리 속도가 고속
    : 컴퓨터 기종 간에 호환성이 없다.

  - 어셈블리어
    : 기계어와 1:1로 대응시켜 사용자가 이해하기 쉽도록 이진 코드를 기호화한 언어
    : 기계어보다 작성이 쉽지만, 컴퓨터 기종 간에 호환성이 없어 전문 프로그래머만 사용

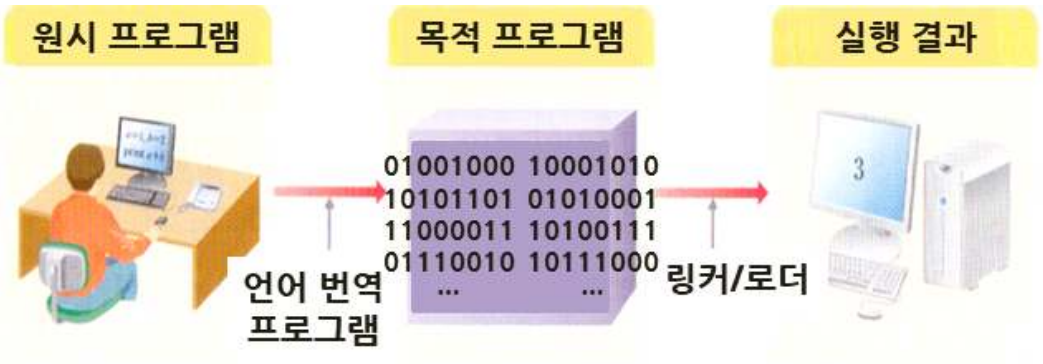


③ 고급언어(High level Language)

- 일상생활에서 사용하는 자연어와 유사하기 때문에 배우기 쉽고 쓰기 쉬운 언어로 컴퓨터 기종과 관계없이 활용할 수 있다.

- 언어 번역 프로그램

: 고급언어는 기계어로 번역되기 위해서는 언어 번역 프로그램이 필요하다.

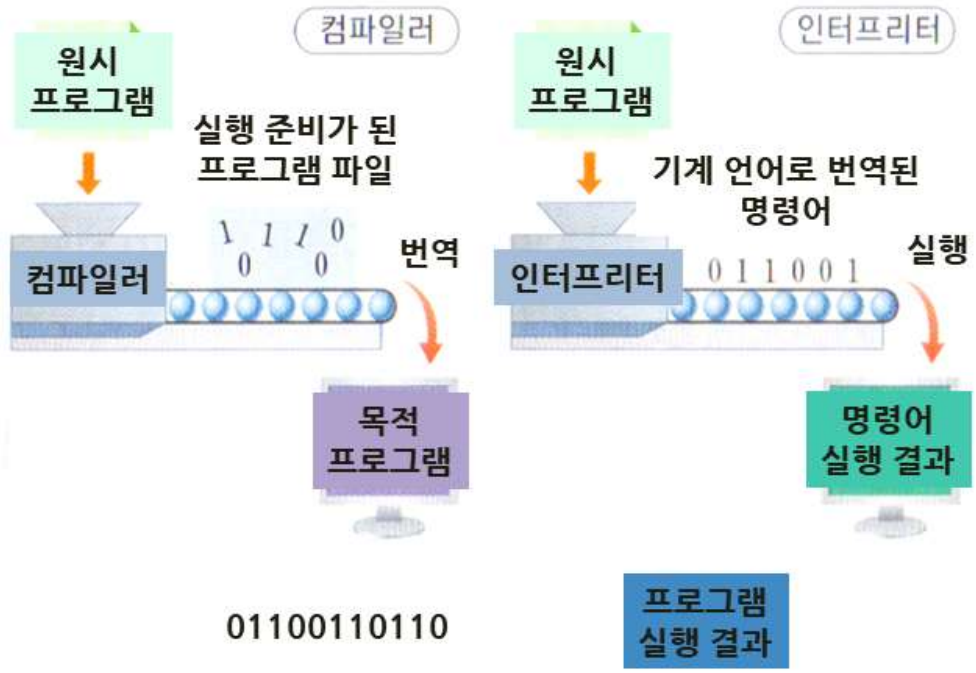


: 원시 프로그램을 목적 프로그램으로 번역하기 위해서는 언어 번역 프로그램(컴파일러, 인터프리터)이 필요

: 목적 프로그램을 실행 가능한 기계어로 번역해주기 위해서는 링커와 로더가 필요

- 컴파일러

: 고급언어(C, C++, C#, Java 등)로 작성한 원시 프로그램 전체를 목적 프로그램으로 번역해 준다.



.

- 인터프리터

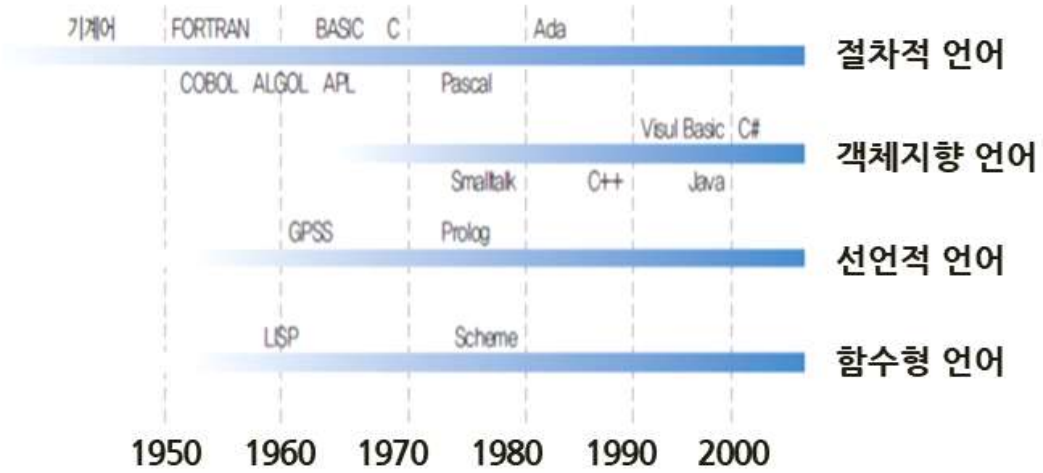
- 원시 명령문 단위로 하나씩 번역하여 즉시 실행(목적 프로그램을 생성하지 않음)

학습내용2 : 프로그래밍 언어의 역사와 구조

- 프로그래밍 언어는 변천하여 자연어와 유사하게 발전할 것을 강조

1. 프로그래밍 언어의 역사

1) 프로그래밍 언어의 패러다임은 접근방법을 개념적으로 분류 P-O-D-F
[프로그래밍 언어의 패러다임과 발전과정]



참조: Computer Science: An Overview, J. Glenn Brookshear

2. 프로그래밍 언어의 종류

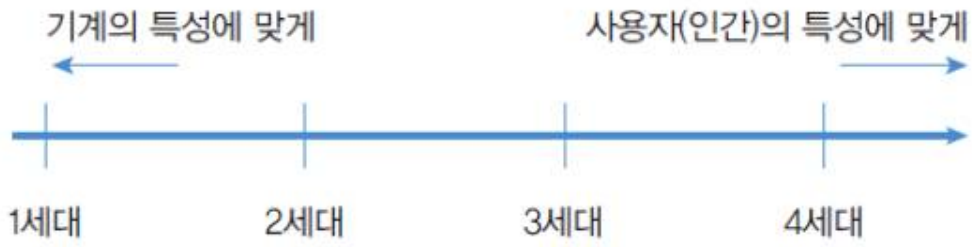
① 절차적 언어(Procedural Programming Language)
- 또는 명령형 언어(Imperative Language), 기본적으로 알고리즘을 표현하기 위한 명령어들의 집합으로 구성
구조적 프로그래밍 개념
- COBOL, FORTRAN, ALGOL, BASIC, PASCAL, C, Ada 등

② 객체지향 언어(Object-Oriented Programming Language)
- 객체(Object)라는 엔티티에 데이터와 메소드(Method)가 포함
- 객체 내의 데이터를 접근하기 위하여 필히 객체의 메소드를 통해 가능

③ 선언적 언어(Declarative Programming Language)
- 절차적 언어는 “문제를 어떻게 풀지(How to solve the problem)”를 기술
- 선언적 언어는 “문제가 무엇인지(What the problem is)”를 정의
- 시뮬레이션 언어 GPSS, 논리적 언어 Prolog, 데이터베이스 질의어 SQL 등

④ 함수형 언어(Functional Programming Language)
- 블록 단위의 프로그램이 마치 수학의 함수와 같이 작용
- 인공지능 언어 LISP, Scheme 등

⑤ 4세대 프로그래밍 언어
- 제 1세대 언어: 기계어
- 제 2세대 언어: 어셈블리어
- 제 3세대 언어: 고수준 프로그래밍 언어
- 제 4세대 언어
    - 초고수준 언어로 선언적 언어의 특성을 가짐
    - How to do(작업의 방법)에서 What to do(작업의 대상) 표현방식의 변화
    - Visual Studio(Visual Basic, Visual C++), Delphi, PowerBuilder, SQL, RPG 등
    - 인공지능언어 : LISP, Prolog
    - 웹 문서 작성 : HTML, XML
    - 사용자 인터랙션 : JavaScript, Perl, Ajax



[프로그래밍 언어의 세대]

3. 프로그래밍 언어의 구조

① 프로그램은 세 가지 유형의 문장
- 선언문 : 변수의 자료형, 상수의 선언
- 명령문 : 프로그램의 알고리즘을 수행할 명령어
- 주석

[프로그램의 선언부와 명령부]

② 변수와 자료형(Data Type)

- Integer, float(real), character, Boolean

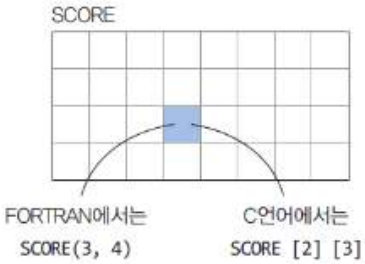
```
float     Length, Width;
int       Price, Tax, Total;
char      Symbol;
```

* C, C++, Java에서 사용

③ 행렬(Array)과 스트럭처(Structure)

- 행렬 : 같은 유형의 데이터가 모인 자료구조
- 레코드 또는 스트럭처 : 다른 유형의 데이터가 모인 자료구조

```
int SCORE [2] [3] ;          ← C
INTEGER SCORE(3, 4)          ← FORTRAN
```




```
struct {char      Name[20];
        int       Age;
        float     SkillLevel;}
        Employee;
```

④ 리터럴(Literal)과 상수(Constant)

```
EffectiveHeight ← Height + 300
LastName ← "Lee"
const int BaseAlt = 500;     ← C
final int BaseAlt = 500;     ← Java
```

⑤ 배정문(Assignment statement)

```
A = X + Y;      ← C, C++, Java
A ← X + Y       ← APL
```

⑥ 제어문 - 조건문과 반복문

- 구조적 프로그래밍(Structured Programming)
- 조건문: if-then-else, switch

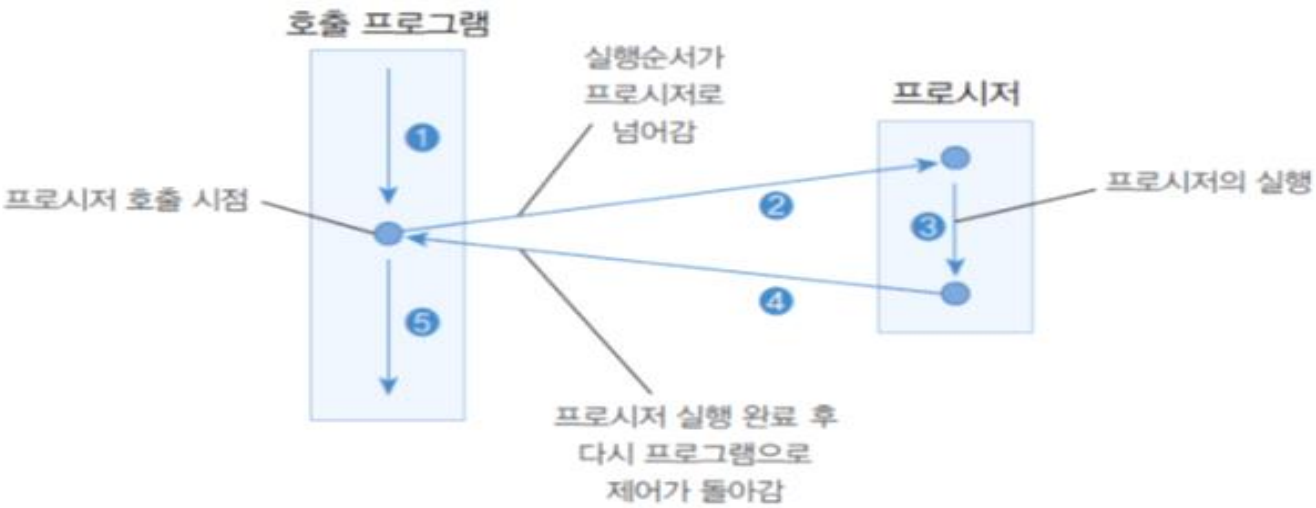
```
if (condition) statementA
        else statementB;

switch (variable) {
        case 'A' : statementA; break;
        case 'B' : statementB; break;
        case 'C' : statementC; break;
        default : statementD}
```

- 반복문: while문, for문

```
while (condition)
        {loop body}

for (int Count = 1; Count < 5;  Count++)
    body ;
```

⑦ 프로시저(Procedure)와 함수(Function)

- 모듈화 프로그래밍
  - 주 프로그램(Main Program), 서브 프로그램(Sub Program)
- 프로시저와 함수
- 전역변수와 지역변수



[프로그램과 프로시저의 실행 순서]

- C언어로 작성된 프로시저 ProjectGDP의 사례

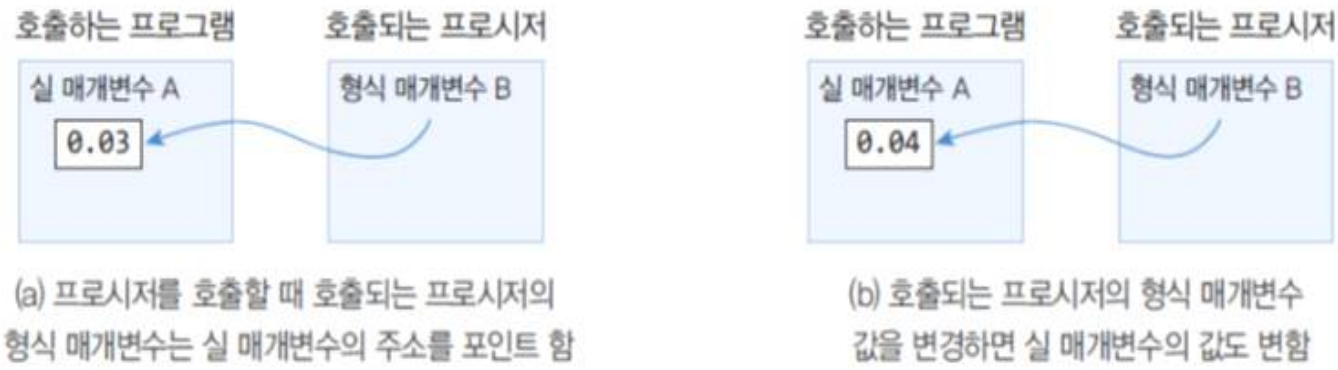
```
C 언어에서 프로시저를 의미함        C 언어로 작성된 프로시저의 형식 매개변수

  void  ProjectGDP  (float GrowthRate)

{ int Year;  ── 지역변수

  GDP[0] = 100.0;
  for (Year = 0; Year =< 10; Year++)
  GDP[Year+1] = GDP[Year] + (GDP[Year] * GrowthRate);
}

                                  행렬 GDP는 전역변수
```

⑧ 매개변수(Parameter)의 값 전달 방식

- 형식 매개변수(Formal Parameter)와 실 매개변수(Actual Parameter) (=실제 데이터 값)
- 프로시저를 호출할 때 매개변수 값을 전달하는 방식
  - 값에 의한 호출(Call by Value)
    : 호출하는 프로그램에서 호출당하는 프로시저로 한번 실 매개변수 값을 전달하면 그것으로 끝남

[매개변수 값에 의한 프로시저 호출]

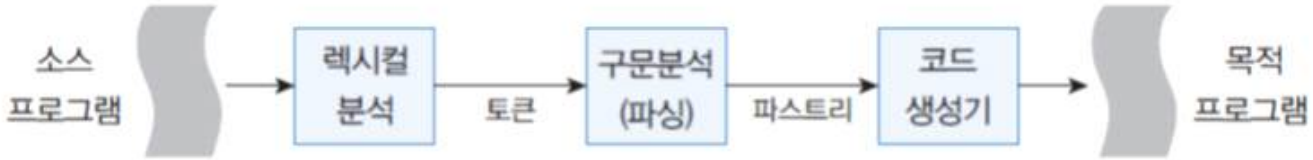
  - 참조에 의한 호출(Call by Reference)
    : 호출하는 프로그램에서 호출당하는 프로시저로 매개변수 값을 전달하는 것이 아니고 매개변수가 저장되어 있는 메모리 주소를 대신 전달



[매개변수 참조에 의한 프로시저의 호출]

4. 프로그래밍 언어의 컴파일 과정

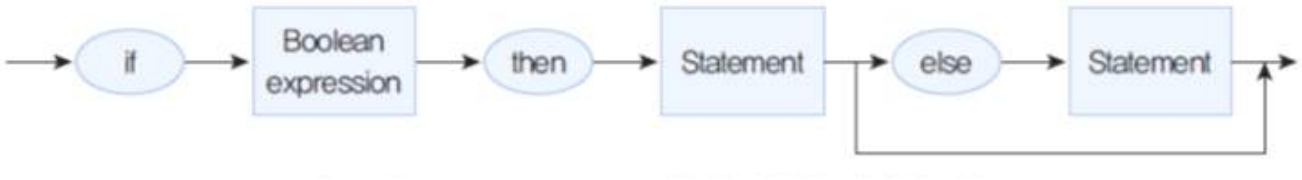
- 렉시컬 분석 과정: 토큰(Token)의 생성
- 파싱(Parsing): 파스트리의 생성
- 코드 생성기: 목적 프로그램의 생성



[컴파일 과정]

① 렉시컬 분석(Lexical Analysis)
- 토큰(Token)의 생성
    - 토큰은 변수 이름, 상수 이름, 리터럴, 예약어
    - 더 이상 분리할 수 없는 프로그램에서 가장 작은 단위

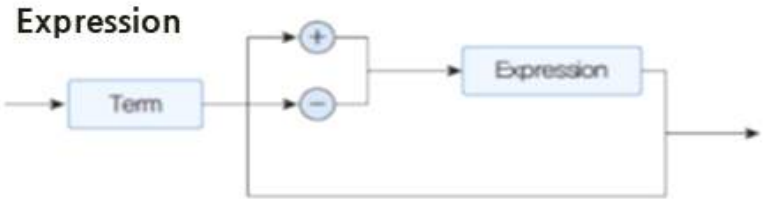
② 구문분석(Syntax Analysis)
- 특정 명령문이 구문 다이어그램으로 표현
    - 명령문을 구문 다이어그램에 맞게 분석하여 하나의 트리 형태로 생성하여 오류검사
    - 사례 : if-then-else 문의 구문 다이어그램



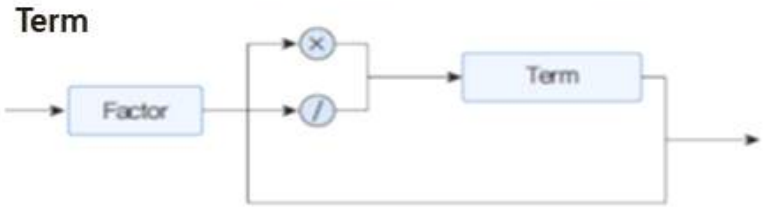
[if-then-else 문장의 구문 다이어그램]

- 터미널(Terminal)과 논터미널(Nonterminal)
- 사례: 배정문에서 “=”의 오른쪽에 오는 표현식(Expression)

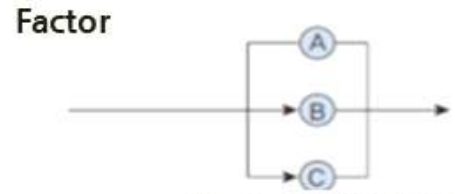
- Term과 factor는 논터미널



(a) Expression의 구문 다이어그램

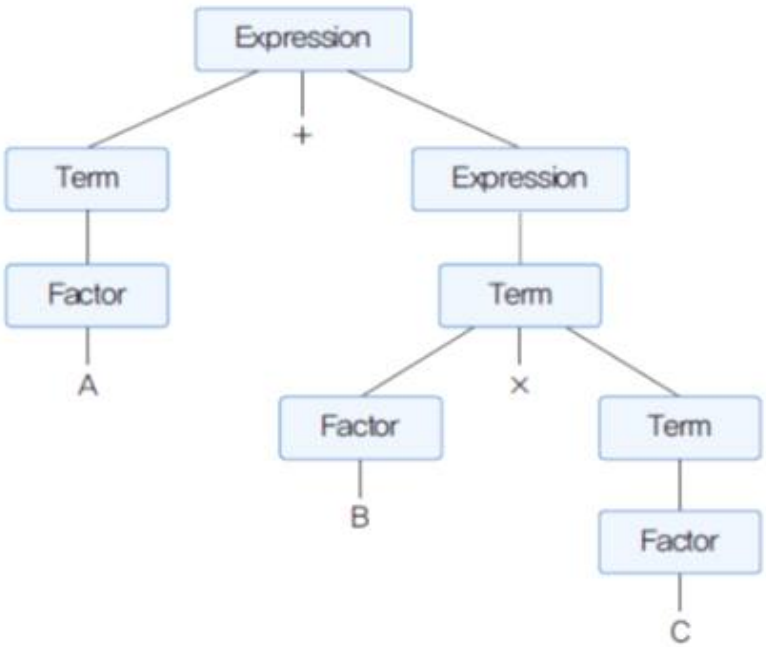


(b) Term의 구문 다이어그램



(c) Factor의 구문 다이어그램

- 사례 : 표현식(Expression) A + B × C의 파싱 결과



③ 프로그래밍 언어의 모호성
- 모호 문법
    - 두가지 이상의 파싱 결과를 초래하는 경우
    - 자연어에서 문법적 모호성이 존재
        예) “He saw a woman in the garden with a telescope.“

【학습정리】

1. 인터프리터(Interpreter)는 미리번역을 해두는 것이 아니라 실행할 때마다 소스프로그램을 한 명령어씩 기계어로 해석하여 바로 실행하는 융통성을 강조한 언어이다.

2. 절차적 언어는 명령형 언어라고도 불라는데 프로그램이 기본적으로 알고리즘을 표현하기 위한 명령어들의 집합으로 구성된다.

3. 절차적 언어에서는 작업의 방법(How to Do)을 표현하지만, 선언적 언어에서는 해야 할 작업의 대상(What to Do)을 표현하는 방식으로 질의어 SQL과 보고서 작성용 RPG가 대표적 언어이다.

4. 값에 의한 호출(Call by Value)은 호출하는 프로그램에서 호출당하는 프로시저로 한번 실 매개 변수 값을 전달하고 끝내는 방식이다.