# 프로그래밍 언어

## 1장 서론

- 프로그래밍 언어란 무엇인가?
- 프로그래밍 언어에서의 추상화
- 프로그래밍 파라다임
- 프로그래밍 언어의 정의
- 프로그래밍 언어의 번역

# 1.1 프로그래밍 언어란 무엇인가 ?

## 프로그래밍 언어란 무엇인가 ?

- 프로그래밍 언어 정의 1
  *컴퓨터에게 우리가 하고 싶은 내용을 전달하기 위한 표기법*
  *man-to-machine communication method*

- 프로그래밍 언어 정의 2
  *프로그래밍 언어는 기계-판독가능(machine-readable)하고 인간-판독가능(human-readable)한 형태로서 계산을 서술하기 위한 표기 체계*
  *not only man-to-machine communication method*
  *but also man-to-man communication method*

## 프로그래밍 언어란 무엇인가?

- 계산(Computation)
  - 컴퓨터가 실행할 수 있는 일련의 과정들
  - 데이터 조작
  - 텍스트 처리
  - 정보 저장 및 검색
- 기계 판독성(Machine readability)
  - 효율적인 번역 혹은 실행
- 사람 판독성(Human readability)
  - 프로그래밍 편의성
  - 컴퓨터 연산들의 이해하기 쉬운 추상화 혹은 요약

## 1.2 프로그래밍 언어에서의 추상화

# 추상화

▣ 자동차로 알아보는 추상화

| | |
|---|---|
| 자동차 운전자 | • 자동차의 상세한 구성이나 엔진의 작동 원리를 알 필요가 없이 기본 동작 원리만 익혀 손쉽게 운전할 수 있음 |
| 자동차 정비사 | • 엔진의 구조, 연료 호스나 냉각 파이프 등과 같은 부품과 자동차의 전체적인 구조에 대해 잘 알아야 함 |
| 자동차 제조업체 | • 복잡한 내부 구조는 주요 부품의 모듈 단위로 차체 내부에 숨겨놓고 운전하는데 반드시 필요한 중요 몇 가지 동작 기능들만 대시보드에 노출 시켜 운전자가 조작할 수 있도록 함 |

7

# 추상화

▣ 추상화(디테일 숨기기, abstraction)란?

- 불필요하고 복잡한 세부적인 내용들은 숨기고 반드시 필요한 몇 가지 핵심적인 특징만으로 단순하게 사물을 이해하고 사용하는 과정

▣ 추상화 장점

- 사용자에게 반드시 필요한 기능들만 남겨놓고 실제로 필요하지 않는 세부 내용들은 모두 숨겨 놓기 때문에 사용자는 간단히 정리된 내용만 알면 됨
- 크고 복잡한 문제들을 단순하게 정리하여 사용하고 관리할 수 있음

8

# 추상화

▣ 동물의 계층 구조로 알아보는 추상화

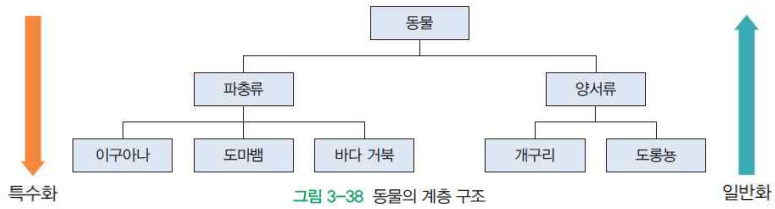



그림 3-38 동물의 계층 구조

- 동물은 하위 계층에 속하는 파충류와 양서류의 공통된 속성을 포함
- 두 번째 계층의 속성인 파충류는 이구아나, 도마뱀, 바다 거북으로 세분화 됨
- 상위 계층에서 하위 계층으로 내려 가면서 속성이 세분화되는 것을 특수화(specialization)라고 함
- 상위 계층으로 올라가면서 이구아나, 도마뱀, 바다 거북의 파충류와 개구리와 도롱뇽의 양서류가 동물이라는 공통적 특징으로 단순화 되는데, 이것을 일반화(generalization)라고 함

9

# 소프트웨어의 추상화

▣ 자동차에 탑재된 소프트웨어의 계층 구조로 알아보는 추상화


그림 3-41 자동차에 탑재된 소프트웨어의 계층 구조

10

# 소프트웨어의 추상화

▣ 자동차에 탑재된 소프트웨어의 계층 구조로 알아보는 추상화

- 자동차의 주행 동작은 크게 3 계층으로 나눌 수 있는데,

| | |
|---|---|
| 센서 측정 계층 | 센서나 카메라를 통해 환경을 인식하는 것으로, 사람의 눈과 같은 역할 |
| 판단 프로그램 계층 | 인지한 상황에 대한 정보를 바탕으로 주행 전략을 판단하는 것으로, 사람의 두뇌와 같은 역할 |
| 제어 프로그램 계층 | 실질적으로 차량을 제어하는 것으로 사람의 혈관이나 근육에 해당하는 역할 |

11

# 소프트웨어의 추상화

▣ 자동차에 탑재된 소프트웨어의 계층 구조로 알아보는 추상화

- 도로 위에서 주행중인 자동차의 충돌 회피 기능은 어떻게 동작할까?

① 차량 거리 센서가 전방 차량의 거리를 측정하여 거리 판단 프로그램으로 결과값을 전달
② 차량의 거리 판단 프로그램은 전달받은 거리 값으로 거리를 판단하여 충돌 여부를 판단
③ 충돌 위험이 있을 경우 거리 판단 프로그램은 제동 제어 프로그램과 안전 제어 프로그램에 위험 상황을 알려줌
④ 제동 제어 프로그램은 운전자 계기판에 위험 정보를 알려준다. 운전자는 계기판 경보 정보의 도움으로 위기 상황을 인지하고 감속 운전을 함

12

## 소프트웨어의 추상화

▣ 자동차에 탑재된 소프트웨어의 계층 구조로 알아보는 추상화

- 거리 센서의 측정 값을 기준으로 계층별 여러 프로그램의 작동을 거치는 복잡한 과정을 수행하더라도 운전자는 계기판의 경보 정보만 확인한다.
- 충돌 회피 기능에 참여하는 각 계층별 프로그램과 센서 정보는 모두 숨겨지고, 운전자는 반드시 필요한 기능인 계기판의 충돌 회피 경보를 확인하고 안전 운전을 한다.

13

## 1.2 프로그래밍 언어에서의 추상화

## Abstraction

- 추상화 과정의 두 가지 고려 점
  - hiding the details
  - showing the important parts

## Abstractions in programming

- Abstraction 분류
  - data abstraction
    - 데이터의 성질을 추상화
  - control abstraction
    - 루프, 조건문등과 같이 프로그램의 실행 경로를 수정
- Abstraction Levels
  - 추상화에 포함되는 정보의 양
  - 기본적 추상화
  - 구조적 추상화
  - 단위적 추상화

## 데이터 추상화(Data Abstraction)

- **기본적 추상화**
  - 일반적인 데이터 값들의 컴퓨터 내부 표현을 추상화
  - **변수(variable)**
    - 데이터 값을 저장하는 메모리 위치
  - **데이터 타입(data type)**
    - 값들의 종류에 대한 이름
    - 예: int, float, double, …
  - **선언(declaration)**
    - 변수의 이름과 데이터 타입을 선언한다.
    - int x;  double y;

## 데이터 추상화

- **구조적 추상화**
  - 관련된 데이터 값들의 모음을 요약
  - **예**
    - 레코드(구조체): 다른 타입의 값들의 모음
    - 배열: 같은 타입의 값들의 모음

## 제어 추상화(Control Abstraction)

- **기본적 추상화**
  - 몇 개의 기계어 명령어들을 하나의 문장으로 요약
  - 배정문
    - x = x+3

      READ X
      ADD X, 3, TMP
      STORE TMP, X
  - goto 문
    - jump 명령어의 요약

## 제어 추상화

- **구조적 추상화**
  - 프로그램 수행의 흐름을 제어하는 검사 내부에 포함되는 명령문의 집단들
- **예**
  - if-문
  - switch-문 C
  - for, while, ...

## 제어 추상화

- **프로시저(함수, 메소드)**
  - 선언
    - 일련의 계산 과정을 하나의 이름을 부여하고 수행할 동작을 그 이름과 연관되게 정의
  - 호출
    - 이름과 실 매개변수를 이용하여 호출
  - Return 값의 유무에 따라
    - 프로시저와 함수를 구분하기도 함

## 통합 추상화(Unit Abstraction)

- **추상 데이터 타입**
  - (데이터 + 연산)
  - 데이터와 관련된 연산들을 통합하여 요약
- **예**
  - Modula-2의 모듈
  - Ada의 패키지
  - C++, Java 등의 클래스

# 1.3 프로그래밍 페러다임

# 명령형 언어(Imperative language)

- *Imperative programming languages began by imitating and abstracting the operations of a von Neuman model computer*
- **특징**
  - 순차적으로 명령어 실행
  - 메모리 위치를 나타내는 변수 사용
  - 배정문을 사용한 변수 값 변경
- **단점**
  - 언어 필요가 아니고
  - 기계 모델에 기반한 언어

## 함수형 언어(Functional language)

- **기본 모델**
  - 수학 함수에 기반
  - 함수를 값에 적용(application)
  - 매개변수 전달(parameter passing)
  - 반환 값(return value)
- **특징**
  - 변수 및 배정문이 없음
  - 자기호출(recursion)에 의한 반복
  - 루프 같은 반복문은 없음
    - 변수 정의가 없기 때문

## 논리 언어 (Logic language)

- **기본적 모델 및 특징**
  - 기호 논리를 기반으로 함
  - 루프나 선택문 등가 같은 제어 추상화가 없음
  - 제어는 하부 시스템(해석기)에 의해 제공
  - 변수는 메모리 위치가 아니라 부분 결과 값에 대한 이름
- **장점**
  - 기계-독립적이고 정확한 의미구조를 가짐
  - 선언적 프로그래밍이 가능
    - 선언적 언어

# 객체지향 언어 (Object-oriented language)

- **기본 모델**
  - 객체(object)
    - 데이터와 관련 연산들의 모음
  - 계산과정(computation)
    - 객체들 사이의 상호작용
  - 클래스(class)
    - 객체에 대한 타입 정의
    - 객체는 클래스의 한 실례(instance)이다.

# 1.4 언어의 정의

## 구문구조(Syntax)

- 문장을 구성하는 법, 즉 문법.
- C에서의 if-문의 구문 예

  if-문은 단어 `if`뒤 괄호 속에 든 식이 하나 오고, 그 뒤에 문장이 하나 온 뒤, 단어 `else`와 또 다른 하나의 문장으로 이루어진 생략할 수 있는 조건비충족 부분이 따르는 식으로 구성된다

  <if-statement> ::= if (< condition>) <statement>
  [else <statement>]

- 어휘적 구조(lexical structure)와 밀접한 관련
  - 토큰(token)사용 : `if`, `else`...

## 의미구조 (semantics)

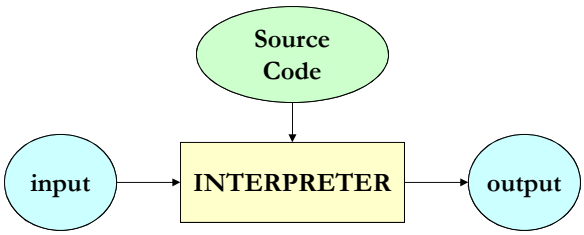
- 문장의 의미, 프로그램의 의미
- 정형적 정의/비정형적 정의
  - if-문의 의미

    if-문은 먼저 식을 평가함으로써 실행을 시작하는데, 그 식은 산술 혹은 포인터 타입을 가져야 한다. 모든 부작용을 포함하여 만약 그 식이 0과 비교하여 같지 않다면, 식을 뒤따르는 문장이 실행된다. 만약 그 식이 0이고 조건비충족 부분이 있으면 `else`를 뒤따르는 문장이 실행된다

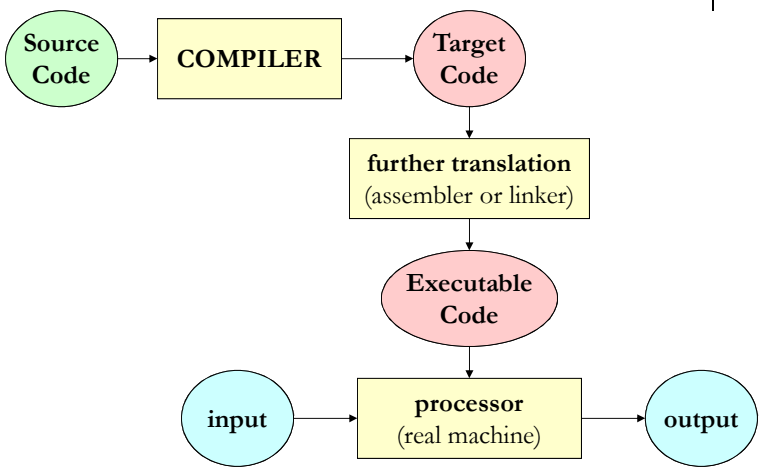
- 정형적 정의 방법
  - 표기적 의미론(denotational semantics)
  - 연산적 의미론(operational semantics)
  - 공리적 의미론(axiomatic semantics)

# 1.5 언어의 번역

# Interpreter



## Compiler




## Pseudo-Interpreter