# 프로그래밍 언어

# 목차

# 학습목표

- 프로그래밍 언어의 기본 개념과 특징을 알아본다.
- 프로그래밍 언어의 발전 과정과 기술 동향을 알아본다.
- 프로그래밍 언어의 종류와 구현 원리를 알아본다.
- 프로그래밍 언어의 실행 과정을 알아본다.
- 절차 지향 언어와 객체 지향 언어의 특성과 사용 절차를 알아본다.
- 교육용 프로그래밍 언어의 특성과 사용 절차를 알아본다.

# 1.1 프로그래밍 언어의 개념

## ❖ 주요 용어

- **프로그램**: 컴퓨터로 문제를 해결하기 위해 작성하는 명령어들의 모임
- **프로그래밍**: 프로그램을 작성하는 과정
- **프로그래머**: 프로그램을 작성하는 사람 또는 직업

## ❖ 프로그래밍 언어

- 프로그램을 작성할 때 사용하는 언어

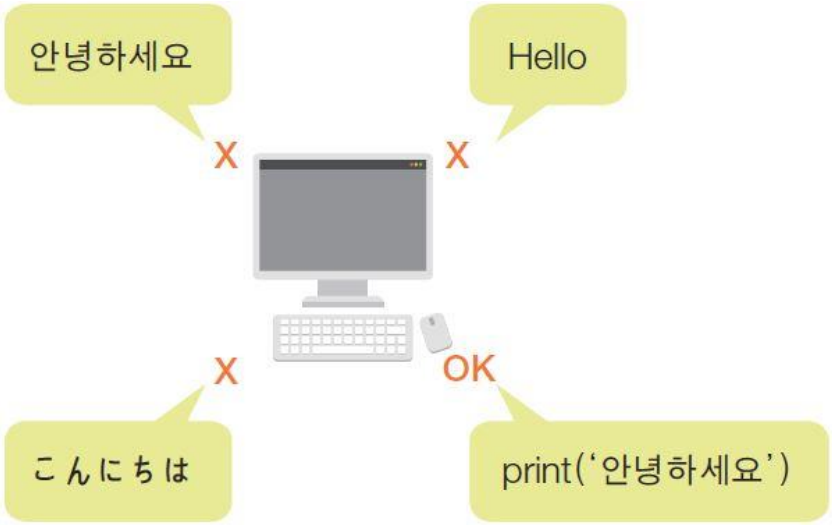
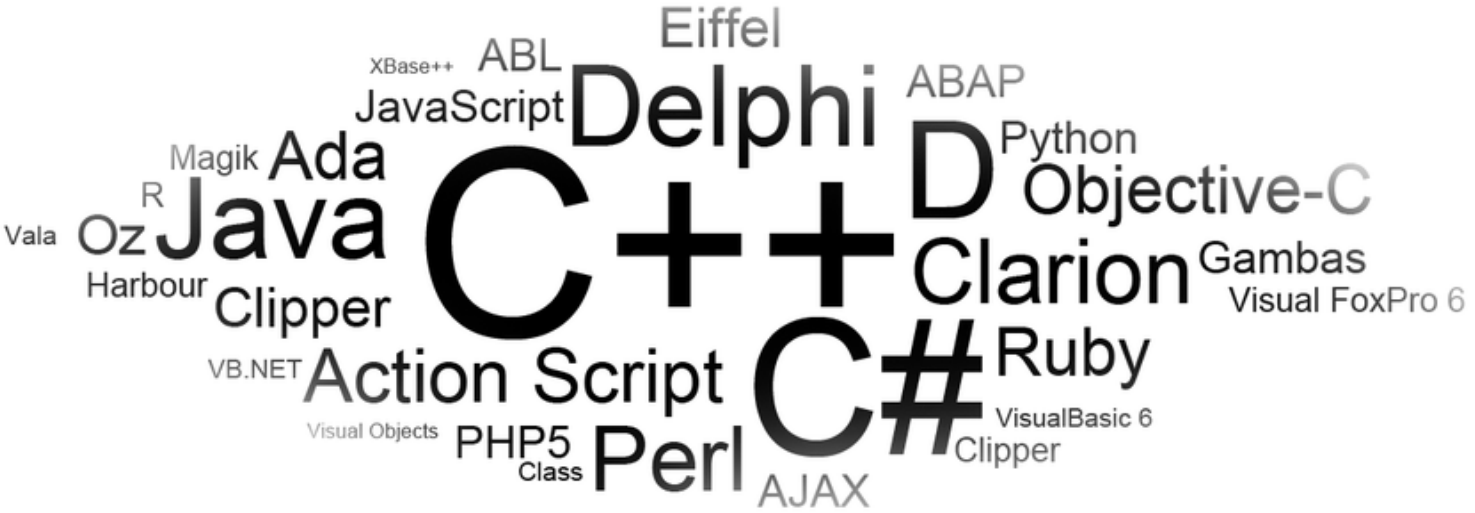



그림 5-1 프로그래밍 언어의 개념

# 1.1 프로그래밍 언어의 개념

## ❖ 프로그래밍 언어

- 인간이 컴퓨터와 의사소통할 수 있도록 컴퓨터에 내리는 명령으로 프로그램을 처리하도록 기술한 언어를 말함



# 1.2 저급 언어와 고급 언어

## ❖ 저급 언어

- 컴퓨터 내부 표현에 가까운 언어로 기계어와 어셈블리어로 구분
- 기계어 : 0과 1로 된 2진수
- 어셈블리어 : 기계어 명령을 알기 쉬운 기호로 표시한 것

# 1.2 저급 언어와 고급 언어

## ❖ 고급 언어

### ▪ 고급 언어의 특징

- ➢ 일상 언어에서 사용하는 표현을 그대로 가져다 쓸 수 있음
- ➢ 사용자가 기억 장소의 주소를 일일이 기억할 필요가 없음
- ➢ 하나의 명령어로 다수의 연산을 실행

### ▪ 고급 언어로 작성한 프로그램을 실행하는 과정

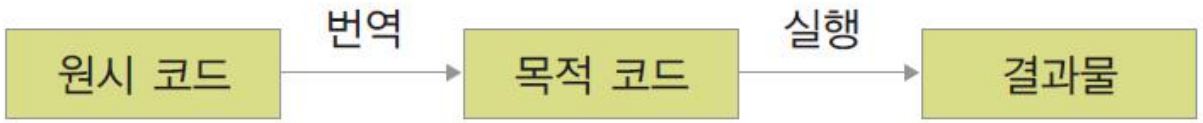



그림 4-1 프로그램 작성과 실행 과정

### ▪ 대표적인 고급 언어

- ➢ C, 포트란, 코볼, 파스칼, C++, 자바, 스몰토크 등

# 1.2 저급 언어와 고급 언어

## ❖ 고급 언어와 저급 언어

- **저급 언어**: 하드웨어 지향의 기계 중심 언어
- **고급 언어**: 사람이 이해하기 쉬운 일상 언어와 기호를 사용한 인간 중심의 언어

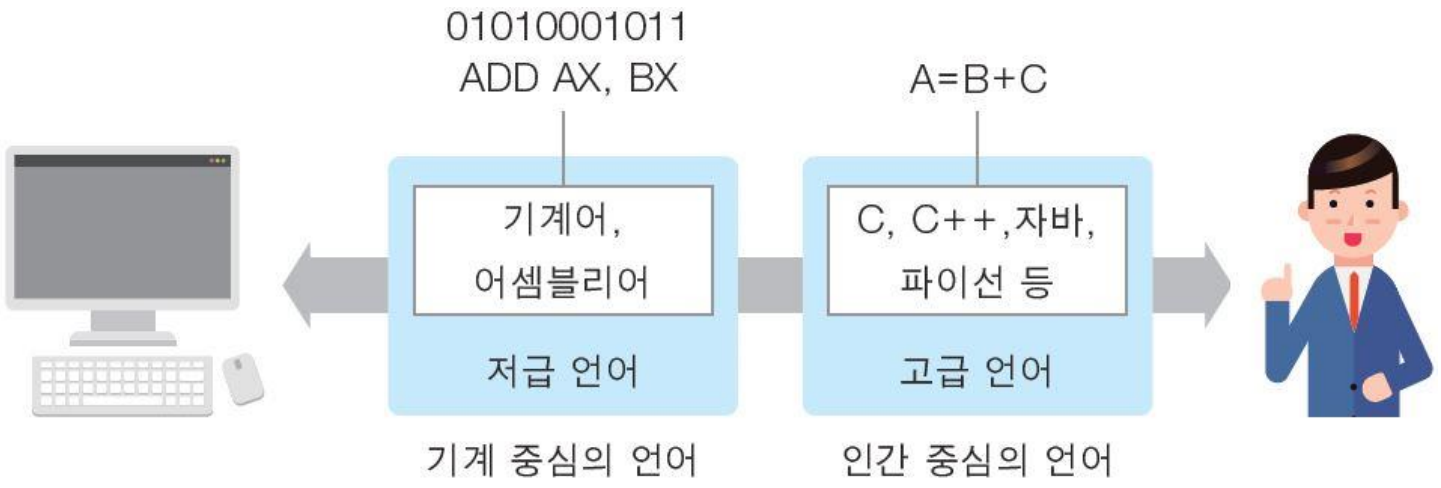



그림 5-2 고급 언어와 저급 언어

# 1.3 프로그래밍 언어의 발전

## ❖ 1950년대 언어

- 포트란 개발 → 프로그래밍 언어 발전의 이정표가 됨

## ❖ 1960년대 언어

- 과학기술용으로 개발된 포트란을 더욱 발전시킨 고급 언어와 사무처리용 고급 언어 출현
- 대표적인 사무처리용 언어 '코볼'

## ❖ 1970년대 언어

- C언어와 파스칼이 개발됨

# 1.3 프로그래밍 언어의 발전

## ❖ 1980년대 언어

- 단말 시스템을 이용한 분산 처리 개념이 확산
- 학생들과 컴퓨터 초보자에게 적합한 교육용 언어가 요구 → 베이직 언어 등장

## ❖ 1990년대 언어

- 1990년대에는 객체 지향 언어가 본격적으로 등장
- C++, 자바, 비주얼 베이직 등의 객체 지향 언어가 새로 등장

## ❖ 2000년대 이후 언어

- 파워빌더, 델파이, 각종 쿼리 전용 언어 등 소위 4세대라 불리는 언어 등장
- 소프트웨어 컴포넌트 기술 발전
- 객체 지향 기술과 웹의 결합을 통해 다양한 정보를 제공하는 기법도 발전
- 최근에는 5세대 언어라 불리는 인공지능 기능을 이용해 자연 언어로 직접 처리하는 기법에 대한 연구가 진행됨

# 1.3 프로그래밍 언어의 발전

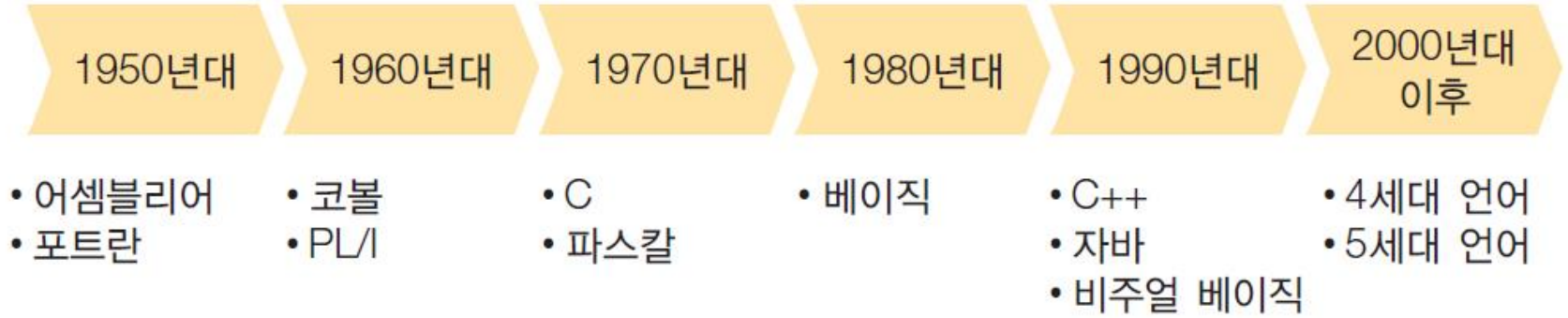



그림 4-2 프로그래밍 언어의 발전 과정

# 1.3 프로그래밍 언어의 발전

## ❖ 프로그래밍 언어의 종류

| 종류 | 특징 | 코드 예 |
|---|---|---|
| C 언어 | • 미국의 벨 연구소에서 데니스 리치(Dennis Ritchie)가 개발한 시스템 프로그래밍 언어다.<br>• 운영체제 등 대부분의 시스템 소프트웨어를 C 언어로 개발한다. | #include <stdio.h><br>int main( )<br>{<br>printf("안녕하세요!");<br>return 0;<br>} |
| 자바 | • 선마이크로시스템사의 제임스 고슬링(James Gosling)이 이끄는 그룹에서 개발했다.<br>• 보안성이 뛰어나며, 인터넷 웹 페이지상에서 실행할 수 있다. | public class Hello<br>{<br>public static void main(String[ ] args)<br>{<br>System.out.println("안녕하세요!");<br>}<br>} |
| 파이선 | • 네덜란드의 프로그래머인 귀도 반 로섬(Guido van Rossum)이 개발했다.<br>• 다양한 플랫폼에서 쓸 수 있고 라이브러리가 풍부하다. | print('안녕하세요!') |

# 1.4 주요 프로그래밍 언어별 특징

## ❖ 코볼

표 4-1 코볼의 장단점

| | |
|---|---|
| 장점 | · 컴퓨터의 내부적인 특성과 별개로 설계되어 코볼 컴파일러만 있으면 컴퓨터 기종에 관계없이 사용할 수 있다.<br>· 파일의 순차 처리와 비순차 처리를 모두 할 수 있어 다른 프로그래밍 언어에 비해 파일 처리 기능이 강력하다.<br>· 작성이 쉽고 이해하기 쉽다. |
| 단점 | · 컴파일러가 많은 항목을 포함하고 있어 주기억장치 용량을 많이 차지한다.<br>· 프로그램 작성량이 많고 길어서 전체적으로 간결하지 못하다. |

표 4-2 코볼 프로그램의 구성

| 디비전 | 설명 | 기술 내용 |
|---|---|---|
| IDENTIFICATION | 프로그램의 내용을 파악하는 식별 디비전 | 프로그램 이름, 작성자, 작성 일자 등 |
| ENVIRONMENT | 프로그램의 처리에 관계되는 환경 디비전 | 컴퓨터 종류, 입출력 파일 및 장치 |
| DATA | 데이터 처리를 위한 기억 장소 디비전 | 기억 장소 형식, 성격, 크기, 내용 등 |
| PROCEDURE | 처리할 명령에 관한 구체적 기술 디비전 | 처리 순서에 따른 명령문 실행의 기술 |

# 1.3 주요 프로그래밍 언어별 특징

## ❖ 파스칼

- **복합문 begin-end, 조건문 if-then-else, 반복문 while-do와 같은 제어 구조가 있어 구조적 프로그래밍에 적합**

```
procedure squareroots (input,output)
var
x : real;
begin
repeat
read(x);
if x ≥ 0
then write(sqrt(x))
else write (‘argument error’)
until x = 0
end
```

# 1.3 주요 프로그래밍 언어별 특징

## ❖ 비주얼 베이직

표 4-3 비주얼 베이직의 장단점

| | |
|---|---|
| 장점 | · 쉽고 간편하게 작성할 수 있어 초보자나 학생들의 교육용으로 사용하기 좋다.<br>· 한글 지원이 우수하다.<br>· 마이크로소프트에서 제공하는 각종 툴을 편하게 이용할 수 있다. |
| 단점 | · 객체 지향 기능이 C#이나 자바 등에 비해 약하다. |

# 1.3 주요 프로그래밍 언어별 특징

## ❖ 비주얼 베이직

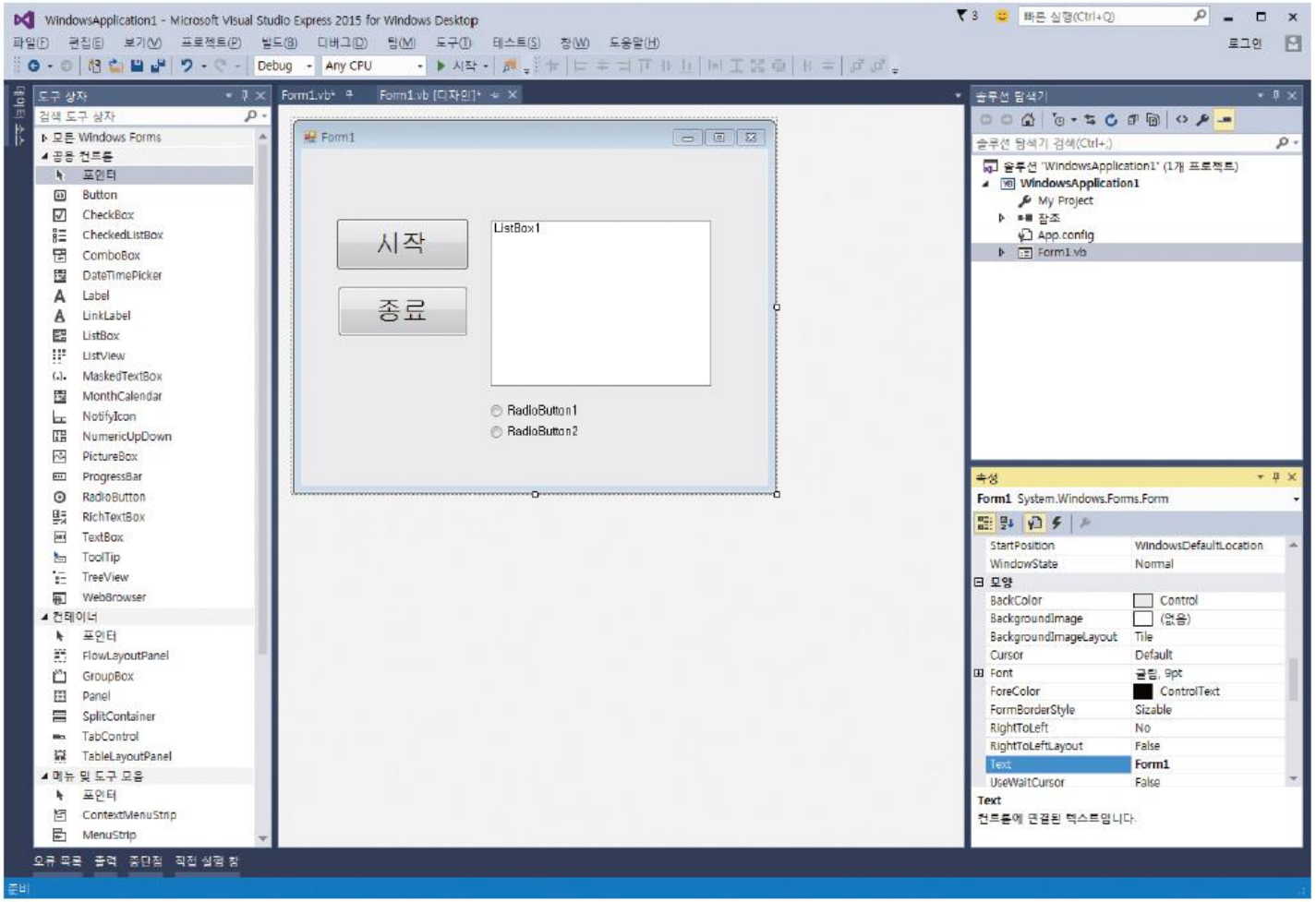



그림 4-3 마이크로소프트 비주얼 스튜디오 2015의 비주얼 베이직 프로그램 화면

# 1.3 주요 프로그래밍 언어별 특징

## ❖ C언어 계열(C, C++, C#)

표 4-4 C언어 계열의 장단점

| | |
|---|---|
| 장점 | · 어셈블리어 같은 저급 언어와 유사한 기능을 포함한다.<br>· 구조적 프로그래밍 기능이 있어 프로그램을 읽고 작성하기 쉽다.<br>· 프로그램의 융통성과 이식성이 상대적으로 뛰어나다.<br>· 기존에 C언어로 개발한 프로그램을 거의 수정하지 않고도 C++로 확장할 수 있어 대부분의 운영체제에서 바로 쓸 수 있다.<br>· 전 세계 수많은 C 프로그래머가 자연스럽게 C++ 프로그래머로 전환할 수 있어 전문 인력이 부족해지는 문제를 해결할 수 있다. |
| 단점 | · C는 객체 지향 개념이 없다.<br>· C++는 방대하고 복잡하여 안정성이 떨어진다. C언어와 호환성이 주요한 특징이므로 새로운 기능을 추가하는 데 한계가 있다.<br>· C#은 자바 사용자 층에 비해 사용자 층이 아직까지 활성화되지 못했다. |

# 1.3 주요 프로그래밍 언어별 특징

## ❖ 자바

- C++의 강력함을 제공하면서도 규모는 더 작고 안전성은 강화된 언어
- 웹 환경에 적합하다는 것이 큰 장점
- 월드 와이드 웹의 보급 확대와 보조를 맞춰 발전

# 1.4 프로그래밍의 이해

## ❖ 언어 번역 프로그램


## ❖ 컴퓨팅 사고력

- 컴퓨터 과학의 이론, 기술, 도구를 활용하여 현실의 복잡하고 어려운 문제를 해결하는 사고 방식

## ❖ 파이선

- 네덜란드의 귀도 반 로섬이 개발한 프로그래밍 언어
- 오픈 소스로 공개되어 있어 무료로 사용할 수 있고, 문법이 쉬워 빠르게 배울 수 있음

# 2.1 사용자 요구 사항 분석과 프로그램 설계

## ❖ 사용자 요구 사항 분석

- 사용자의 필요를 파악하고 프로그램을 통해 해결할 문제가 무엇인지 확인하는 단계

## ❖ 프로그램 설계

- 실제 코딩을 시작할 때 사용할 논리를 프로그래머가 대략 그려내는 단계
- 알고리즘 설계라고도 함

- 알고리즘의 특성
  - 알고리즘 명령을 수행하면 유한한 횟수를 거친 후 종료해야 한다.
  - 알고리즘의 각 단계와 명령은 명확하게 정의되어야 한다.
  - 알고리즘은 데이터 입력이 0 또는 그 이상이어야 한다.
  - 알고리즘은 한 가지 이상의 결과를 출력한다.
  - 알고리즘은 효과적이어야 한다. 이는 유한한 시간 내에 정확히 수행할 수 있을 정도로 단순해야 함을 의미한다.

# 2.1 사용자 요구 사항 분석과 프로그램 설계

## ❖ 프로그램 설계

- 프로그램 제어 흐름 유형
  - 순차 구조 : 프로그램 코드 순서대로 실행
  - 선택 구조 : 프로그램이 다음에 무엇을 해야 하는지를 결정하는 분기 구조
  - 반복 구조 : 조건이 만족하지 않을 때까지 계속 반복

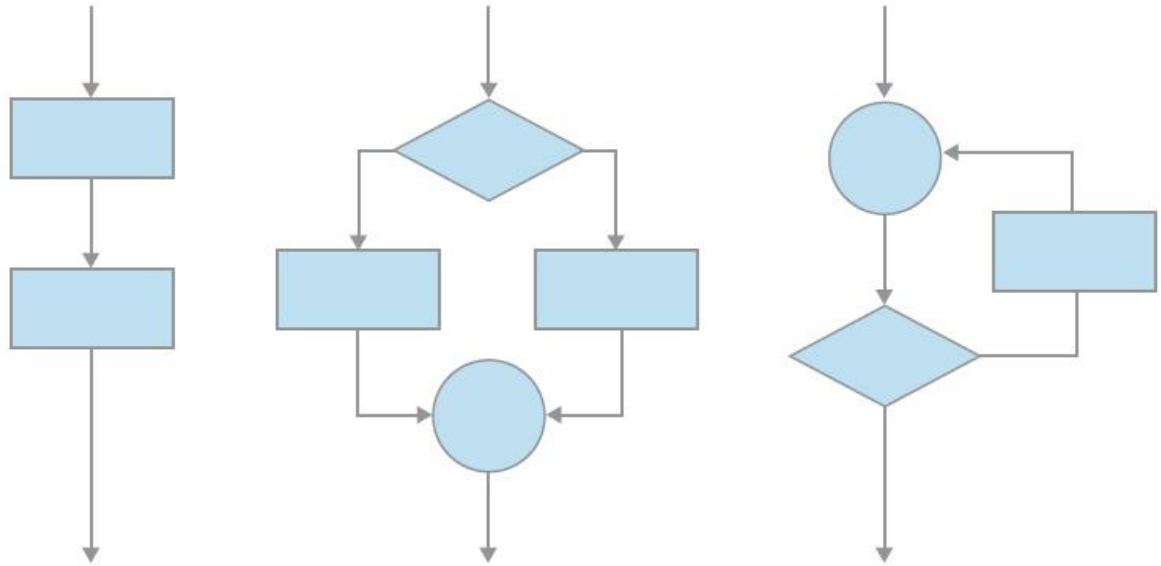



그림 4-4 프로그램 제어 흐름의 유형

# 2.1 사용자 요구 사항 분석과 프로그램 설계

## ❖ 프로그램 설계

### ▪ 반복 구조

#### ➢ For문

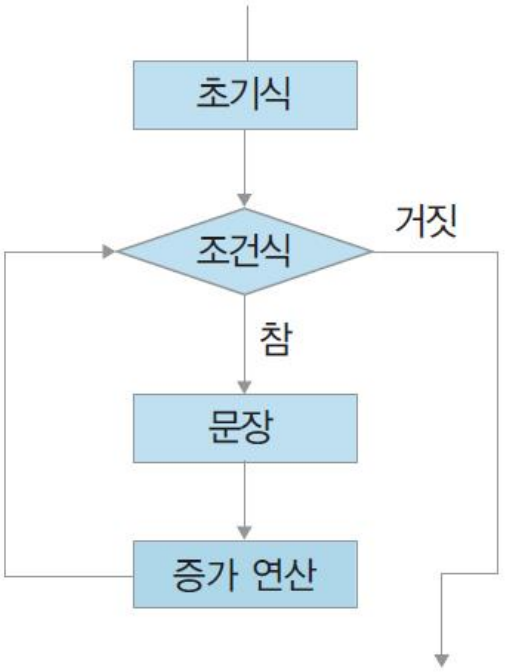



그림 4–5 for 문의 순서도와 명령 형태

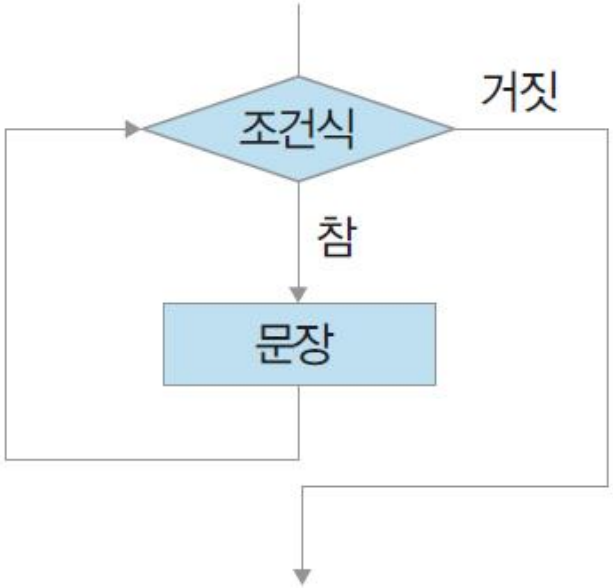
```
for (초기식; 조건식; 증가 연산) {
 문장;
}
```

# 2.1 사용자 요구 사항 분석과 프로그램 설계

## ❖ 프로그램 설계

### ▪ 반복 구조

➢ while문




```
while(조건식) {
 문장;
}
```

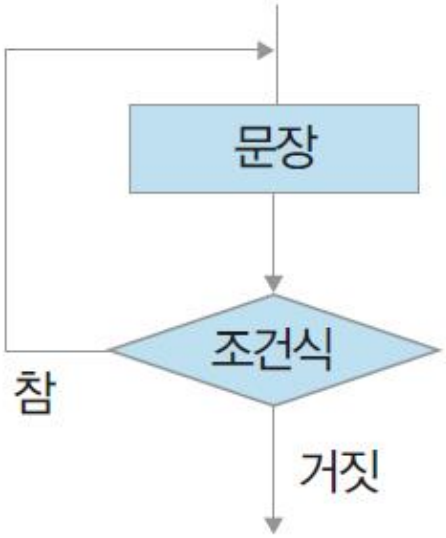
그림 4-6 while 문의 순서도와 명령 형태

# 2.1 사용자 요구 사항 분석과 프로그램 설계

## ❖ 프로그램 설계

### ▪ 반복 구조

➢ **do-while문**




```
do {
   문장;
} while(조건식);
```

그림 4-7 do-while 문의 순서도와 명령 형태

# 2.2 코딩 및 컴파일

❖ 코딩 : 프로그래밍 언어로 프로그램을 작성하는 단계

❖ 컴파일 : 고급 언어로 작성된 명령문을 기계어로 바꾸는 단계

❖ 컴파일러를 이용한 방식

- 프로그램 전체를 한번에 기계어로 번역하는 방식
- C언어, 코볼, 포트란, 파스칼 등의 언어에서 사용

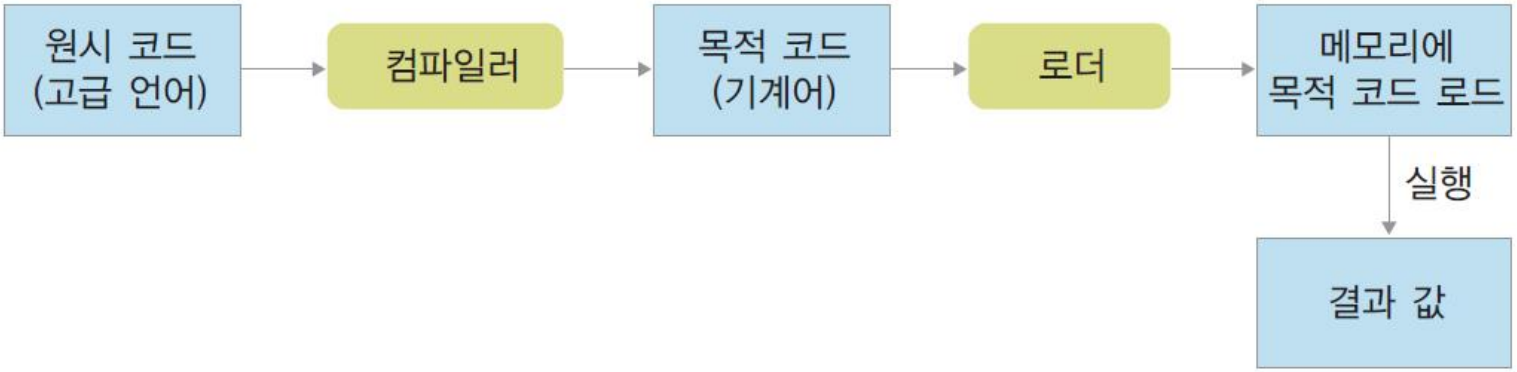



그림 4-8 컴파일러를 이용한 방식

# 2.2 코딩 및 컴파일

❖ 인터프리터를 이용한 방식

- 프로그램을 한 행씩 읽어 번역과 실행을 동시에 하는 방식
- 베이직 등의 언어에서 사용

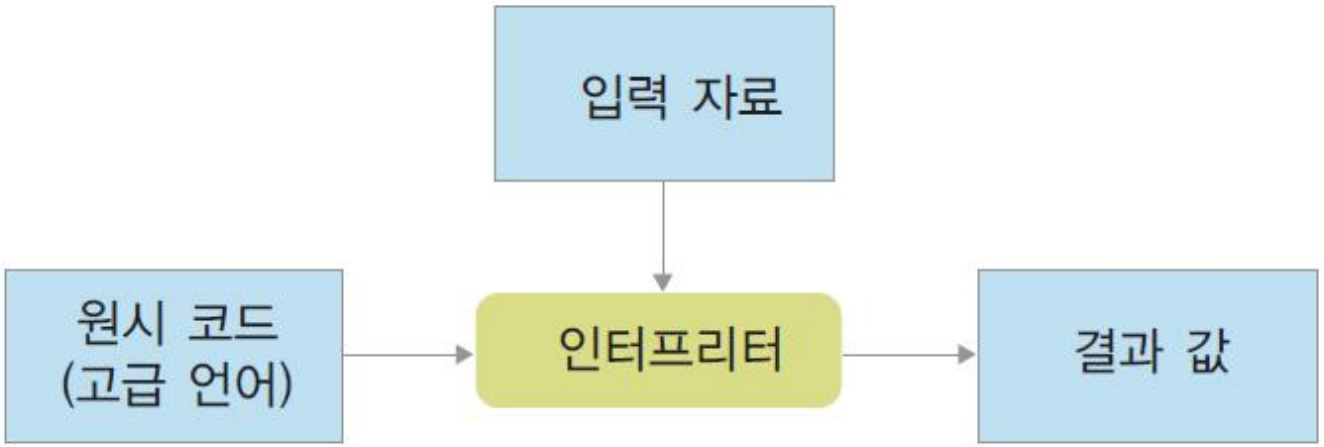



그림 4-9 인터프리터를 이용한 방식

# 2.2 코딩 및 컴파일

## ❖ 하이브리드 방식

- 컴파일러와 인터프리터를 함께 이용하는 방식
- 리스프, 스노볼4, APL, 프롤로그, 자바 등의 언어에서 사용

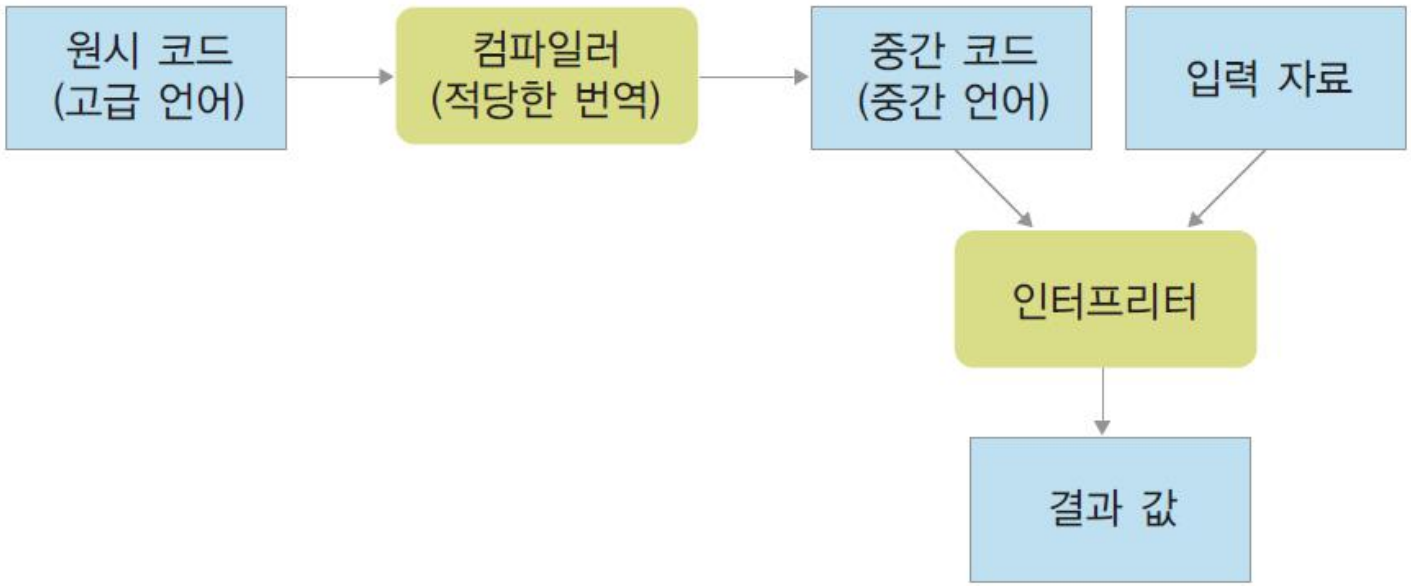



그림 4-10 하이브리드 방식

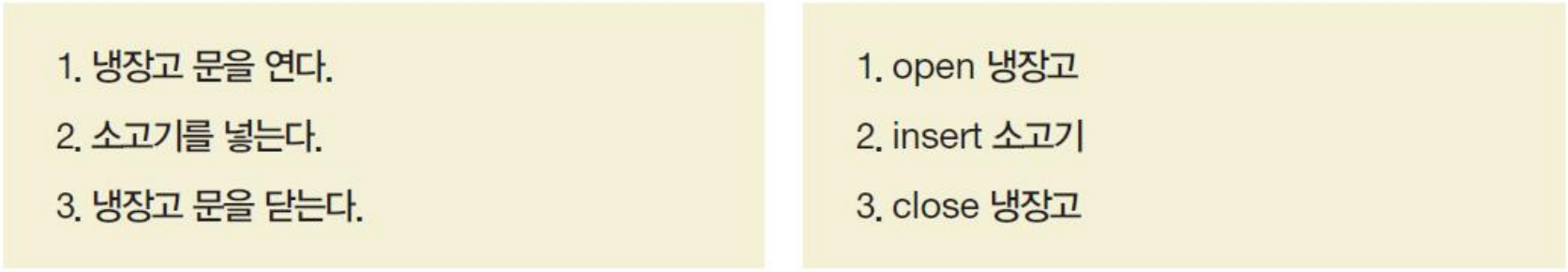
# 2.4 디버깅 및 시험

- 프로그램이 포함하는 모든 오류를 찾아내 제거하는 것
- 오류에는 구문 오류와 논리 오류가 있음
  - 구문 오류 : 틀린 문자를 입력하거나 문법에 맞지 않는 명령문을 사용했을 때 발생하는 오류
  - 논리 오류 : 제어 구조의 부적절한 사용으로 발생하는 오류
- 시험은 알파 테스트와 베타 테스트로 구분
  - 알파 테스트 : 완성된 프로그램을 개발 환경에서 시험하는 방법
  - 베타 테스트 : 특정 고객이 고객에 쓰는 환경에서 시험하는 방법

# 3.1 절차 지향 언어의 개념

## ❖ 절차 지향 언어의 개념

- 프로그램 코드가 순서대로 실행되는 언어
- 파스칼, 코볼, 포트란, 베이직, C언어 등



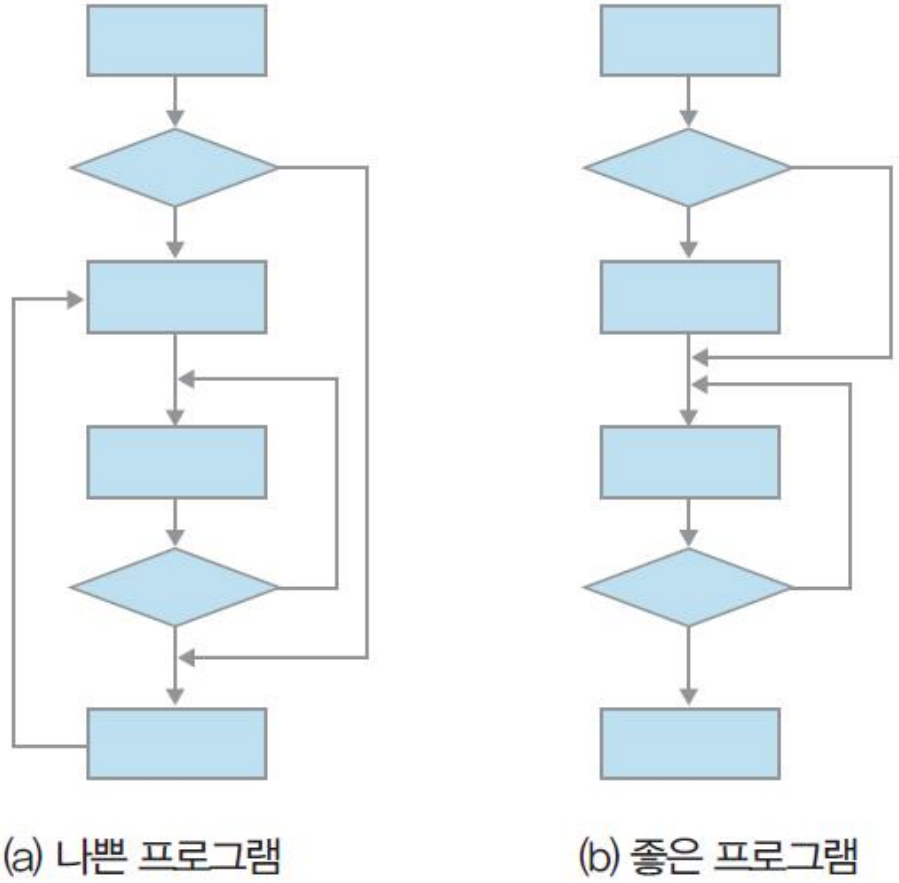

(a) 냉장고에 소고기를 넣는 과정 (b) 냉장고에 소고기를 넣는 프로그램

그림 4-18 절차 지향 언어의 프로그래밍 개념

# 3.2 구조적 프로그래밍의 이해

## ❖ 구조적 프로그래밍 등장 배경

- goto문의 무분별한 분기 구조를 개선하고 모든 명령문의 처리를 블록으로 모듈화시키기 위해 등장

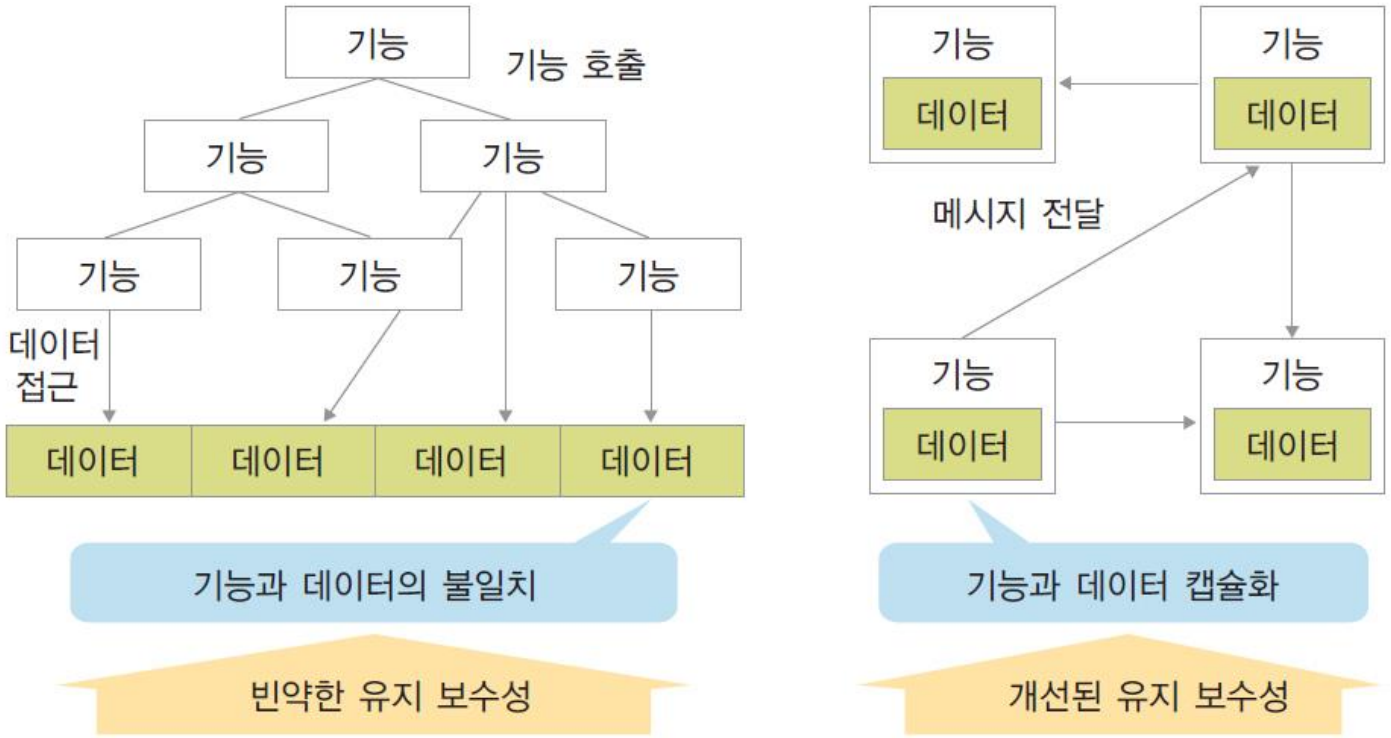


(a) 나쁜 프로그램 (b) 좋은 프로그램

그림 4-19 구조적 프로그래밍의 예

# 4.1 절차 지향 언어와 객체 지향 언어의 차이점

- ❖ 절차 지향 언어 : 데이터와 데이터를 처리하는 기능이 별도로 관리
- ❖ 객체 지향 언어 : 데이터와 기능을 묶어 캡슐화시킨 후 메시지를 전달하여 일을 처리




(a) 절차 지향 언어 (b) 객체 지향 언어

그림 4-20 절차 지향 언어와 객체 지향 언어의 차이점

# 4.2 객체 지향 언어의 주요 개념

## ❖ 클래스

- 다른 사물과 구분되는 속성을 가진 객체가 모여, 일반화된 범주로 묶인 것

## ❖ 객체

- 개별적으로 식별되는 사물을 지칭
- 속성과 기능을 캡슐화 함

## ❖ 상속

- 하위 클래스는 상위 클래스가 가지는 속성과 기능을 모두 이어받을 수 있는데 이를 상속이라고 함

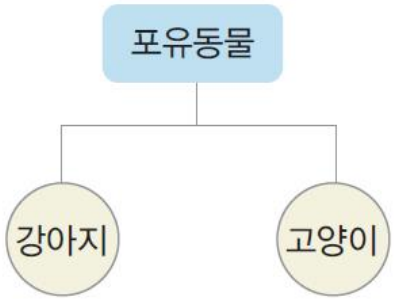



그림 4-22 상속의 예

# 4.2 객체 지향 언어의 주요 개념

## ❖ 메시지

- 객체 간에 전달되는 명령 단위


그림 4-23 객체 간의 메시지 송수신

# 4.2 객체 지향 언어의 주요 개념

## ❖ 추상화

- 어떤 객체가 상대하는 다른 객체에 대해, 꼭 필요한 부분만 알고 나머지 세부적인 사항은 감추는 것

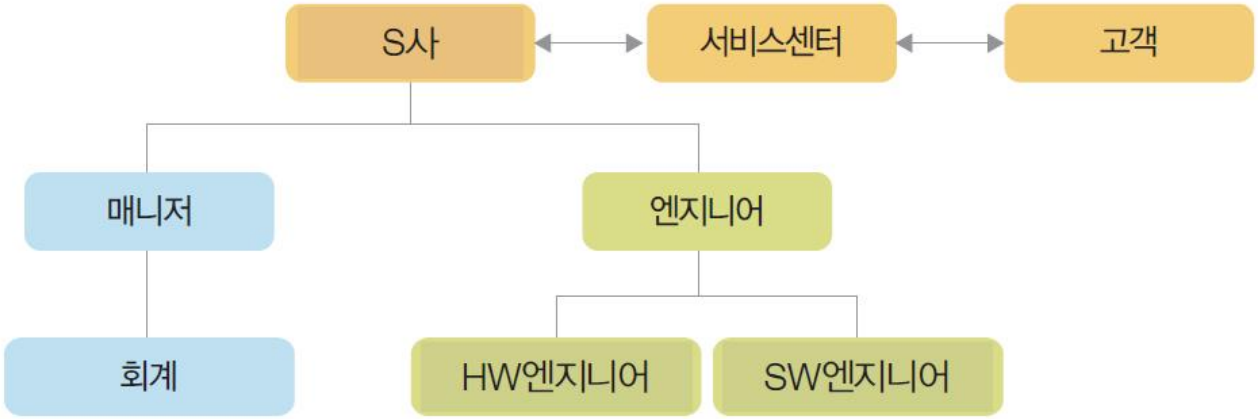



그림 4-24 추상화 개념 사례

## ❖ 캡슐화

- 객체에 속성과 기능을 포함하면서 추상화 개념을 통해 객체의 세부내용은 사용자로부터 은폐하는 것

# 4.2 객체 지향 언어의 주요 개념

## ❖ 다형성

- 일반화된 클래스는 어떤 특정화된 클래스 객체를 지칭할 수 있기 때문에 같은 동작을 함, 하지만 각각 특정화된 클래스는 다른 성질을 가질 수 있음
- 이런 성질을 다형성이라 함

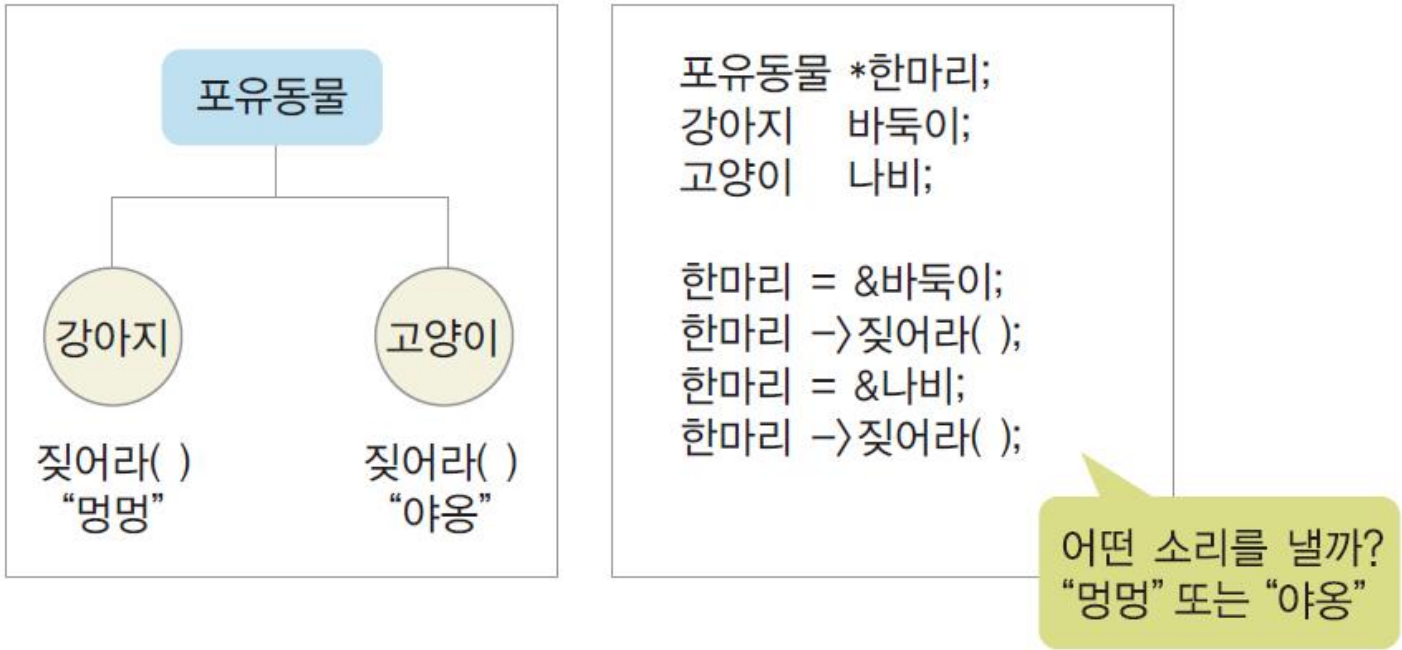



그림 4-25 다형성의 예