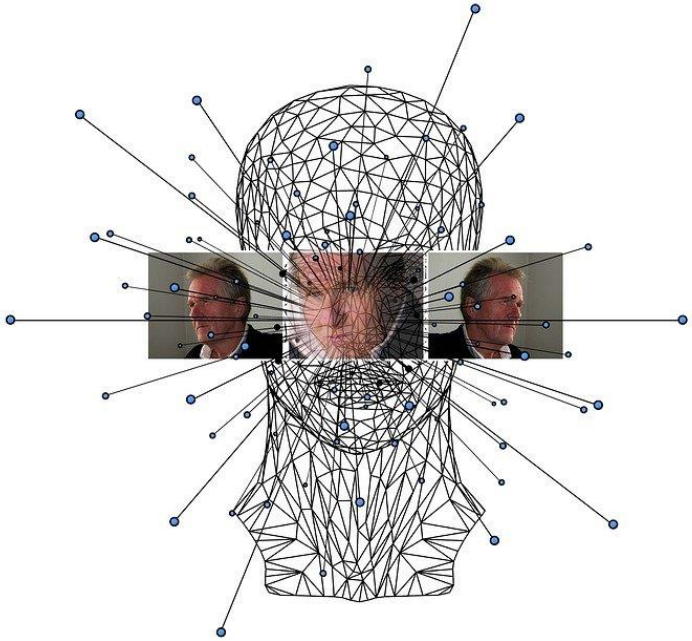
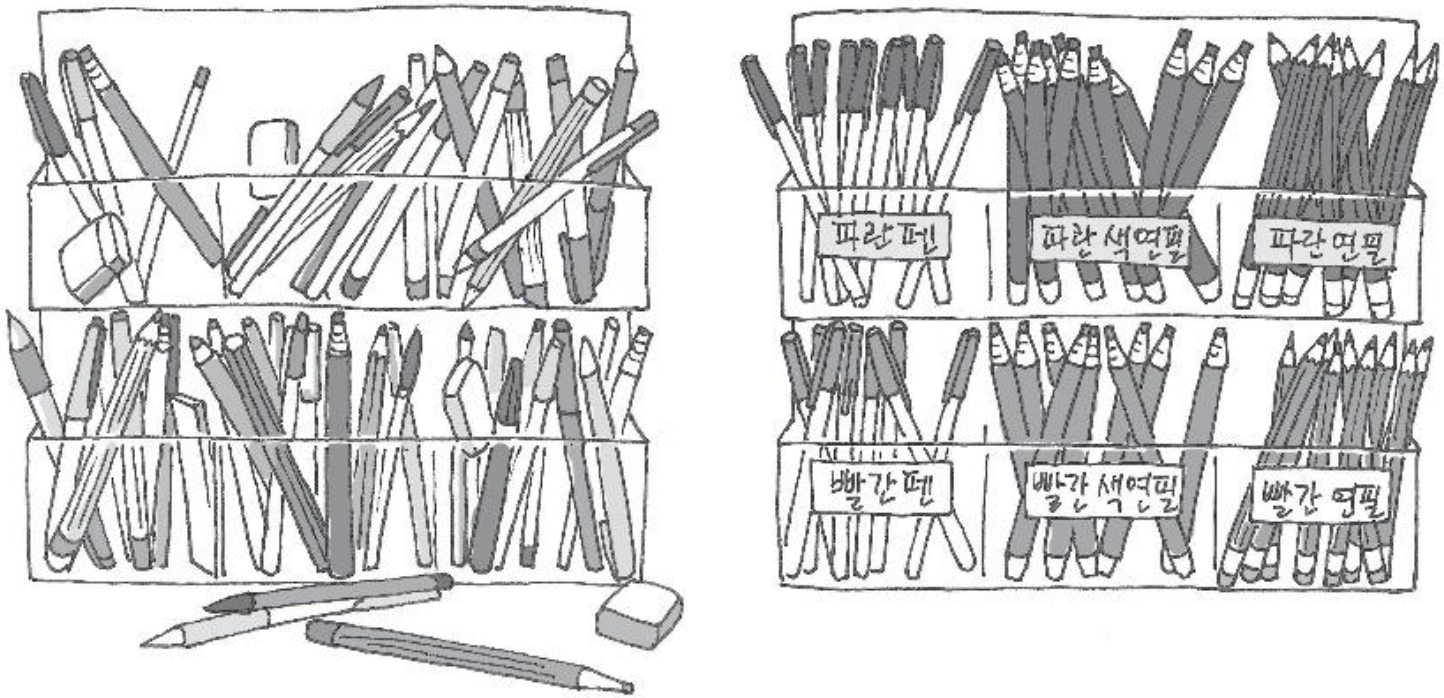


1 자료구조와 알고리즘 소개

1. 자료구조의 이해 : 개념

❖ 자료구조의 개념

- 자료를 효율적으로 표현하고 저장하고 처리할 수 있도록 정리하는 것



(a) 자료구조를 적용하기 전

(b) 자료구조를 적용한 후

그림 1-1 생활 속에서 자료구조를 적용한 예

1. 자료구조의 이해 : 개념

❖ 컴퓨터 분야에서 자료구조를 왜 배워야 하는가?

- 컴퓨터가 효율적으로 문제를 처리하기 위해서는 문제를 정의하고 분석하여 그에 대한 최적의 프로그램을 작성해야 한다.
  - 자료구조에 대한 개념과 활용 능력 필요!

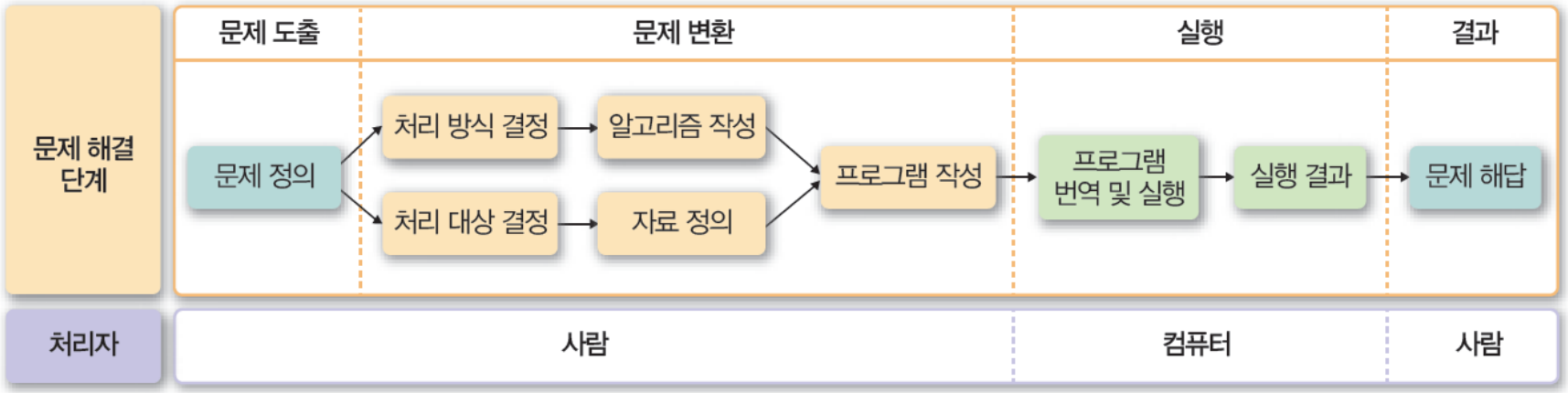


그림 1-2 문제 해결 과정

1. 자료구조의 이해 : 분류

❖ 자료의 형태에 따른 분류

- 단순 구조
  - 정수, 실수, 문자, 문자열, 등의 기본 자료형
- 선형 구조
  - 자료들 사이의 관계가 1:1 관계
  - 순차 리스트, 연결 리스트, 스택, 큐, 데크 등
- 비선형 구조
  - 자료들 사이의 관계가 1:다, 또는 다:다 관계
  - 트리, 그래프 등
- 파일 구조
  - 서로 관련 있는 필드로 구성된 레코드의 집합인 파일에 대한 구조
  - 순차 파일, 색인 파일, 직접 파일 등

1. 자료구조의 이해 : 분류

❖ 자료의 형태에 따른 분류와 이 책에서 다루는 세부 주제

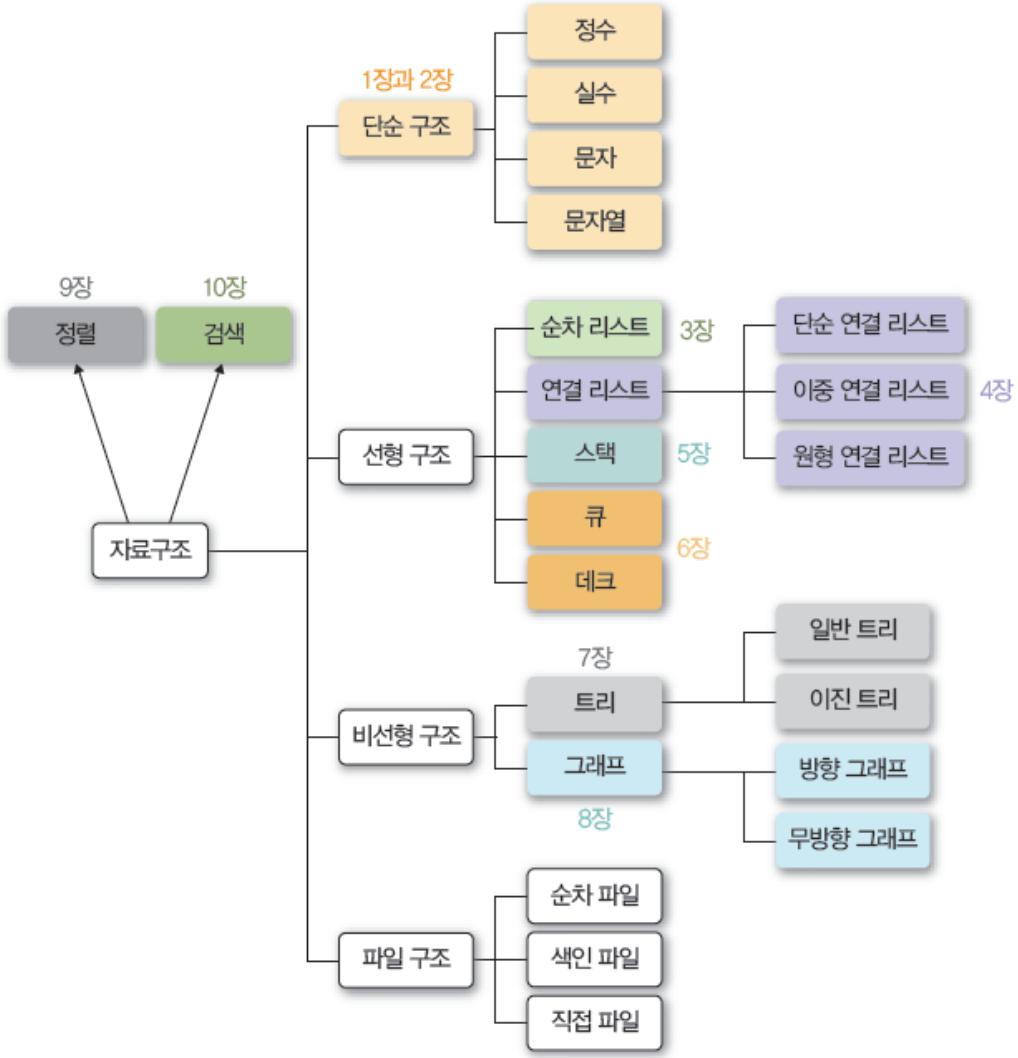


그림 1-4 자료구조의 형태에 따른 분류와 이 책에서 다루는 세부 주제

2. 자료의 표현

❖ 컴퓨터에서의 자료 표현

- 숫자, 문자, 그림, 소리, 기호 등 모든 형식의 자료를 2진수 코드로 표현하여 저장 및 처리
- 2진수 코드란?
  - 1과 0, On과 Off, 참True과 거짓False의 조합
- 2진수 코드의 단위

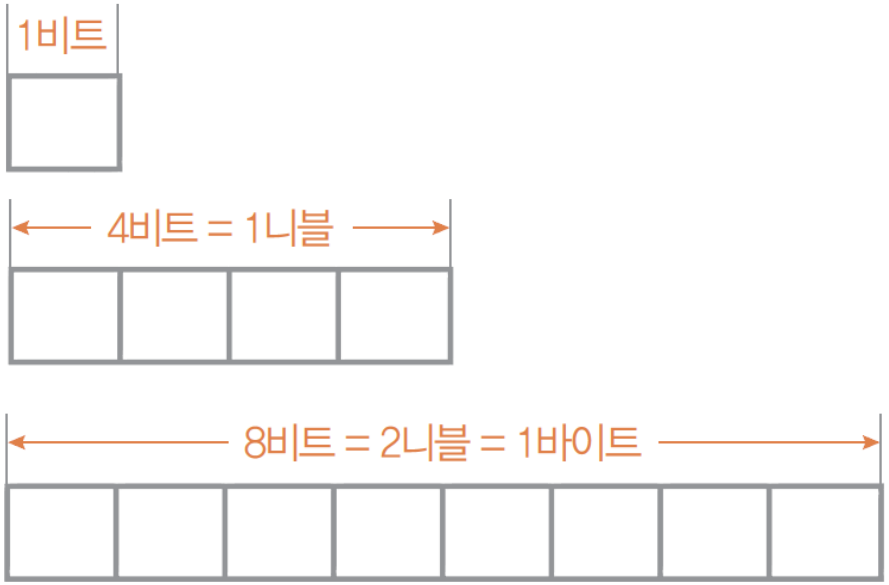
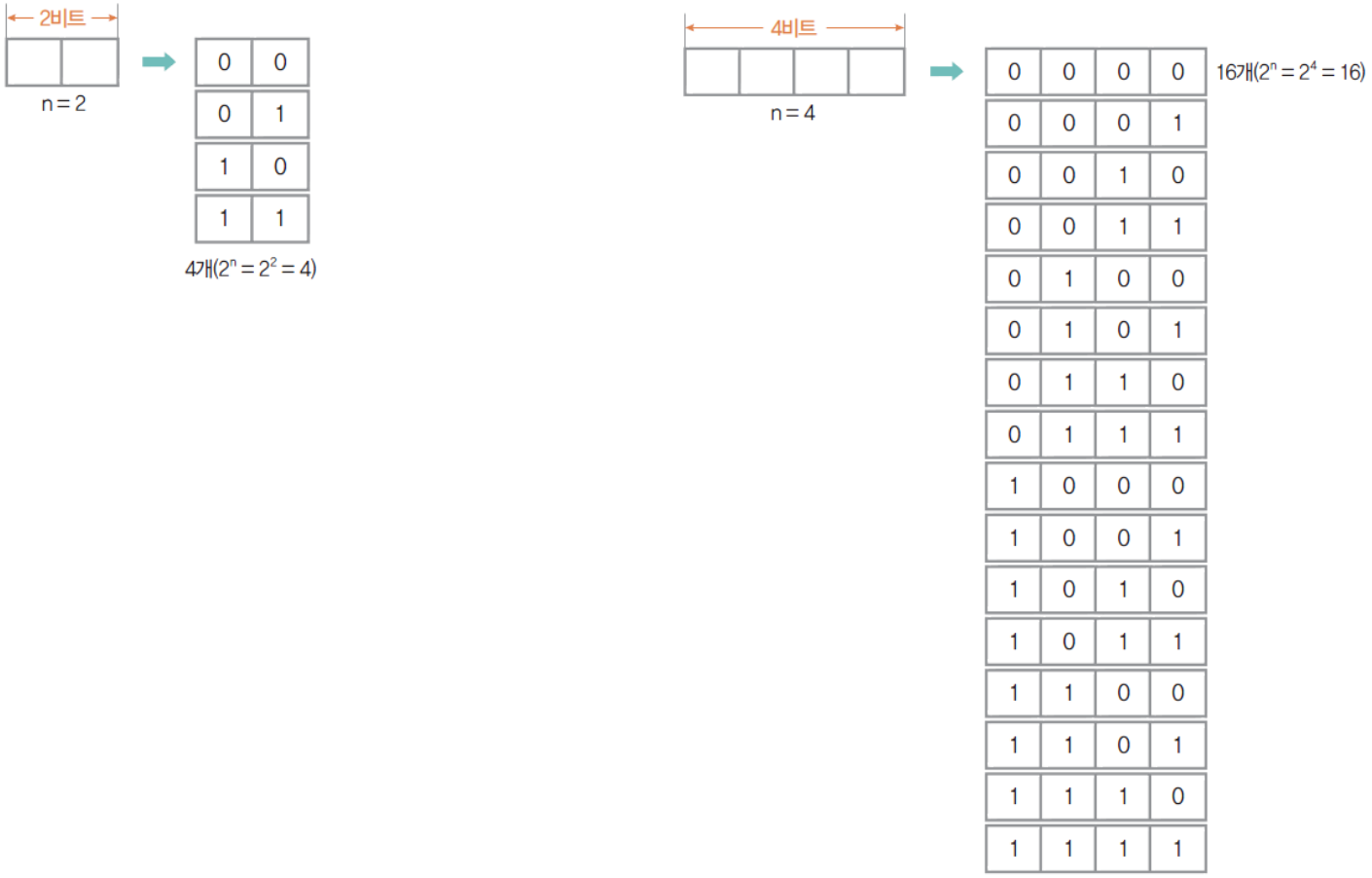


그림 1-5 컴퓨터의 자료 표현 : 비트, 니블, 바이트

2. 자료의 표현

❖ 디지털 시스템에서의 자료 표현

- n개의 비트로 2^n개의 상태 표현



(a) n=2인 경우 : 2^2개의 상태 표현

(b) n=4인 경우 : 2^4개의 상태 표현

그림 1-6 자료 표현 예 : n개의 비트로 2^n개의 상태 표현

2. 자료의 표현

❖ 컴퓨터 내부에서 표현할 수 있는 자료의 종류

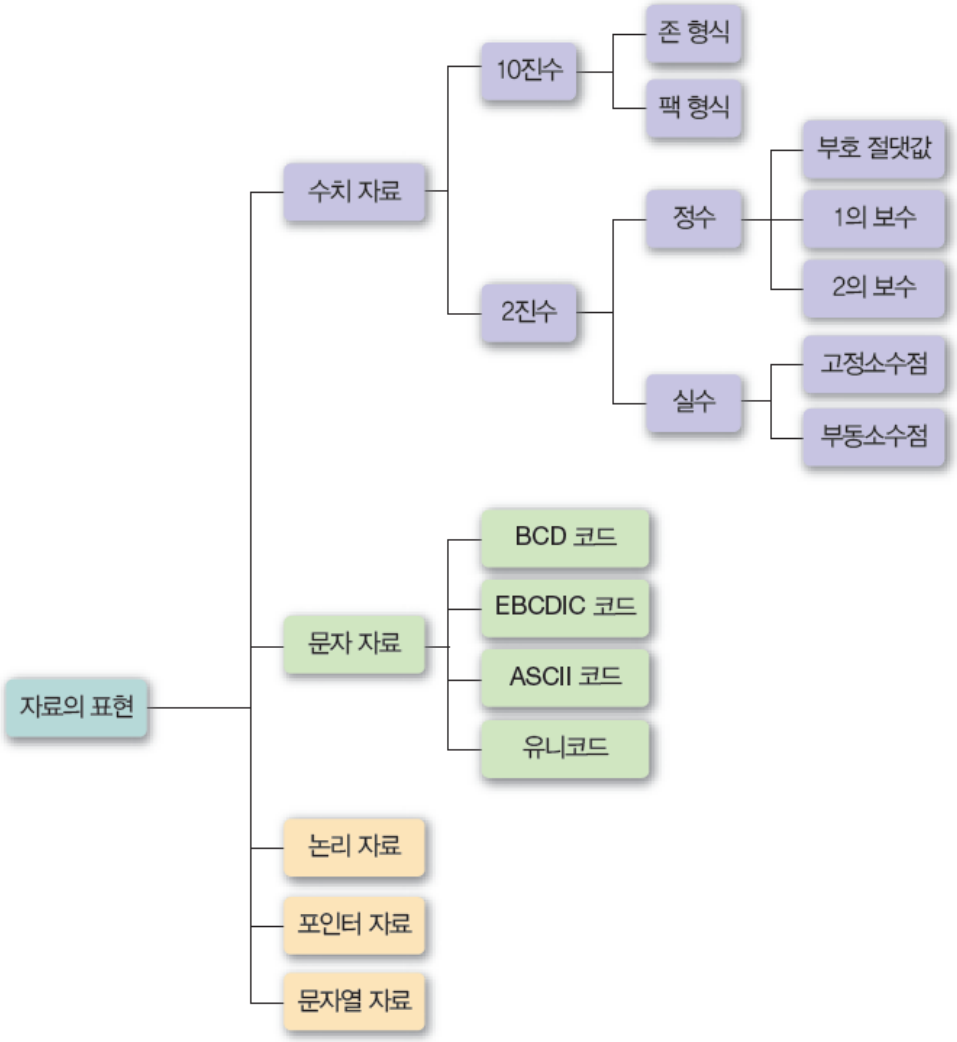


그림 1-7 컴퓨터 내부에서 자료를 표현하는 방법

3. 자료의 추상화

❖ 뇌의 추상화 기능

- 기억할 대상의 구별되는 특징만을 단순화하여 기억하는 기능

그림 1-23 뇌의 추상화 기능

3. 자료의 추상화

❖ 컴퓨터를 이용한 문재해결에서의 추상화

- 크고 복잡한 문제를 단순화시켜 쉽게 해결하기 위한 방법
- 자료 추상화(Data Abstraction)
  - 처리할 자료, 연산, 자료형에 대한 추상화 표현
  - 자료 : 프로그램의 처리 대상이 되는 모든 것을 의미
  - 연산
    - 어떤 일을 처리하는 과정. 연산자에 의해 수행
    - 예) 더하기 연산은 +연산자에 의해 수행
  - 자료형
    - 처리할 자료의 집합과 자료에 대해 수행할 연산자의 집합
    - 예) 정수 자료형

      자료 : 정수의 집합. {…, -1, 0, 1, …}

      연산자 : 정수에 대한 연산자 집합. {+, -, x, ÷, mod}

3. 자료의 추상화 : 개념

❖ 추상 자료형(ADT, Abstract Data Type)

- 자료와 연산자의 특성을 논리적으로 추상화하여 정의한 자료형

❖ 추상화와 구체화

- 추상화 – "무엇(what)인가?"를 논리적으로 정의
- 구체화 – "어떻게(how) 할 것인가?"를 실제적으로 표현

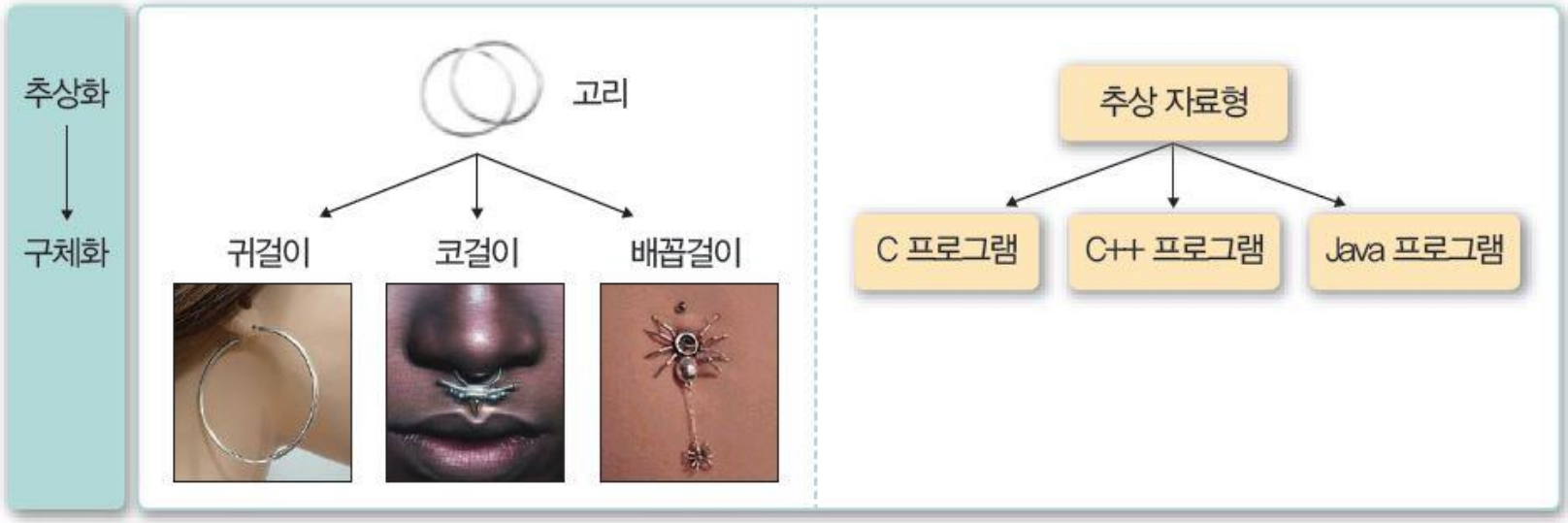


그림 1-25 추상화와 구체화의 예

3. 자료의 추상화

❖ 추상화와 구체화

- 자료와 연산에 있어서의 추상화와 구체화의 관계

표 1-8 자료와 연산의 추상화와 구체화

구분	자료	연산
추상화	추상 자료형	알고리즘 정의
구체화	자료형	프로그램 구현

4. 알고리즘의 이해

❖ 알고리즘

- 문제해결 방법을 추상화하여 단계적 절차를 논리적으로 기술해 놓은 명세서

❖ 알고리즘의 조건

- 입력input : 알고리즘 수행에 필요한 자료가 외부에서 입력으로 제공될 수 있어야 한다.
- 출력output : 알고리즘 수행 후 하나 이상의 결과를 출력해야 한다.
- 명확성definiteness : 수행할 작업의 내용과 순서를 나타내는 알고리즘의 명령어들은 명확하게 명세되어야 한다.
- 유한성finiteness : 알고리즘은 수행 뒤에 반드시 종료되어야 한다.
- 효과성effectiveness : 알고리즘의 모든 명령어들은 기본적이며 실행이 가능해야 한다.

4. 알고리즘의 이해

요리 재료 **자료**

케이크 시트(20cm×20cm) 1개, 크림치즈 무스(크림치즈 200g, 달걀 2알, 설탕 3큰술, 레몬즙 1큰술, 바닐라 에센스 1큰술), 딸기 시럽(딸기 500g, 설탕 1½컵, 레몬즙 1작은술), 딸기 1개, 플레인 요구르트 2큰술

요리법 **알고리즘**

절차

1. 케이크 틀에 유산지를 깔고 케이크 시트를 놓는다.
2. 달걀 2알을 잘 푼다. 볼에 크림치즈를 넣고 거품기로 젓는다. 달걀 푼 물과 설탕 3큰술을 세 차례로 나누어 넣으면서 크림 상태가 되도록 거품기로 젓는다.
3. ❷에 레몬즙과 바닐라 에센스를 넣고 살짝 저은 다음 ❶에 붓는다. 180℃로 예열된 오븐에 전체를 넣고 20분 정도 굽는다.
4. 딸기를 얇게 자르고 냄비에 넣은 다음 설탕 1½컵을 넣고 약한 불로 끓인다. 눌어붙지 않도록 계속해서 젓고 거품이 생기면 걷어 낸다. 되직해지면 레몬즙을 넣고 차갑게 식힌다.
5. 치즈케이크 한 조각을 접시에 담고 ❹를 뿌린 다음 플레인 요구르트와 딸기를 얹는다. **연산**

그림 1-26 딸기 시럽을 얹은 치즈케이크 만들기

4. 알고리즘의 이해

자료

[**요리 재료**]

스펀지케이크(20×20cm) 1개, 크림치즈 200g, 달걀 푼 물 2개 분량, 설탕 3큰술, 레몬즙·바닐라에센스 1큰술씩, 딸기시럽(딸기 500g, 설탕 1½ 컵, 레몬즙 1작은술), 딸기 1개, 플레인 요구르트 2큰술

[**요리법**] >> 알고리즘

절차

① 케이크 틀의 가장자리에 필름을 돌린 다음 스펀지케이크를 놓는다.

② 볼에 크림치즈를 넣고 거품기로 젓다가 달걀 푼 물과 설탕 3큰술을 세번에 나누어 넣으면서 크림 상태로 만든다.

③ ②에 레몬즙과 바닐라에센스를 넣고 살짝 저은 다음 ①에 붓는다. 이것을 180℃의 오븐에 넣고 20분 정도 굽는다.

④ 냄비에 슬라이스한 딸기와 설탕 1½ 컵을 넣고 끓이다가 약한 불에서 눌어붙지 않도록 저으면서 거품을 걷어낸다. 되직해지면 레몬즙을 넣고 차게 식힌다.

⑤ 접시에 치즈케이크를 한 조각 담고 ④의 시럽을 뿌린 다음 플레인 요구르트와 딸기를 얹어낸다

연산

5. 알고리즘의 표현 방법

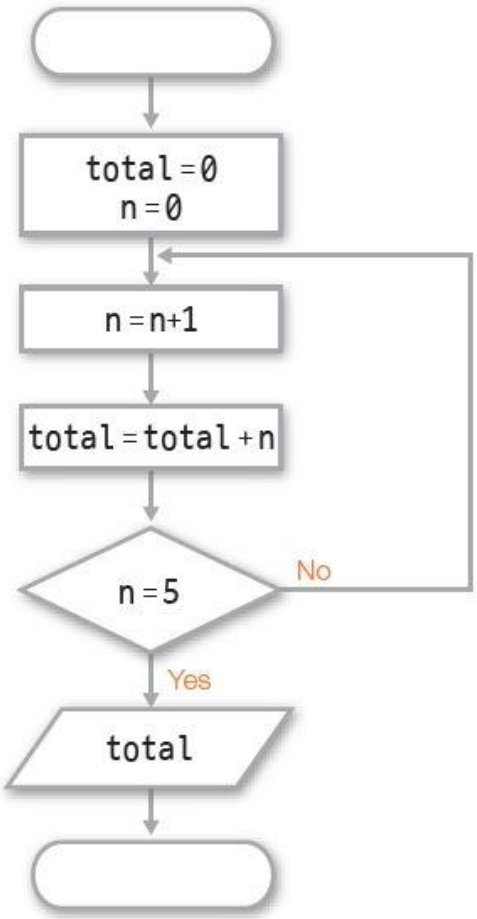
❖ 알고리즘의 표현 방법의 종류

- 자연어를 이용한 서술적 표현 방법
- 순서도Flow chart를 이용한 도식화 표현 방법
- 프로그래밍 언어를 이용한 구체화 방법
- 가상코드Pseudo-code를 이용한 추상화 방법

5. 알고리즘의 표현 방법

❖ 순서도를 이용한 도식화

- 순서도의 예) 1부터 5까지의 합을 구하는 알고리즘



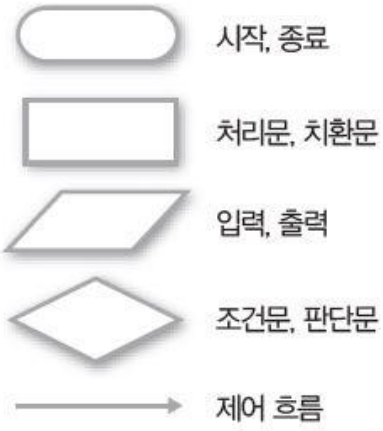


그림 1-27 순서도의 예

5. 알고리즘의 표현 방법

❖ 가상코드를 이용한 추상화

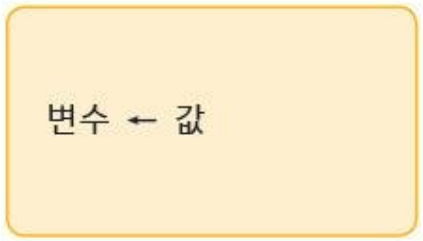
- 가상코드, 즉 알고리즘 기술언어ADL, Algorithm Description Language를 사용하여 프로그래밍 언어의 일반적인 형태와 유사하게 알고리즘을 표현
- 특정 프로그래밍 언어가 아니므로 직접 실행은 불가능
- 일반적인 프로그래밍 언어의 형태이므로 원하는 특정 프로그래밍 언어로의 변환 용이

5. 알고리즘의 표현 방법

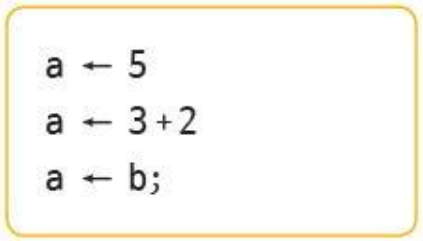
❖ 가상코드의 형식

- 기본 요소
  - 기호
    - 변수, 자료형 이름, 프로그램 이름, 레코드 필드 명, 문장의 레이블 등을 나타냄
    - 문자나 숫자의 조합. 첫 문자는 반드시 영문자 사용.
  - 자료형
    - 정수형과 실수형의 수치 자료형, 문자형, 논리형, 포인터, 문자열 등의 모든 자료형 사용
  - 연산자
    - 산술연산자, 관계연산자, 논리연산자

- 지정문 형식과 예



(a) 지정문 형식



(b) 지정문 예

그림 1-28 지정문 형식과 예

5. 알고리즘의 표현 방식

❖ 조건문

- 조건에 따라 실행할 명령문이 결정되는 선택적 제어구조를 만든다.
- if 문의 형식과 제어흐름

```
if (조건식) then 명령문 1;
else 명령문 2;
```

(a) if – then – else 형

```
if (조건식) then 명령문 1;
```

(b) if – then 형

그림 1-29 기본 if 문의 형식과 제어 흐름

5. 알고리즘의 표현 방식

- 다단계 조건문
  - 중첩 if 문의 형식과 제어 흐름

```
if (조건식 1) then 명령문 1;
else if (조건식 2) then 명령문 2;
else 명령문 3;
```

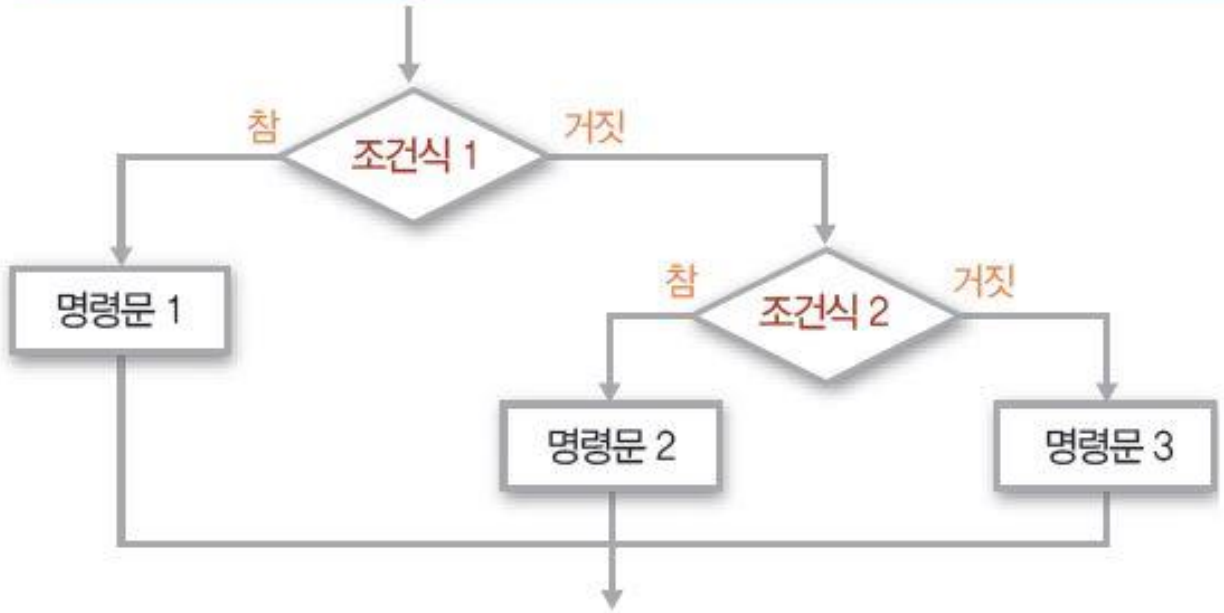


그림 1-30 중첩 if 문의 형식과 제어 흐름

5. 알고리즘의 표현 방식

- 중첩 if문 사용 예) 평균 점수에 따른 등급 계산하기

```
if Average >= 90 then grade ← "A";
else if Average >= 80 then grade ← "B";
else if Average >= 70 then grade ← "C";
else grade ← "F";
```

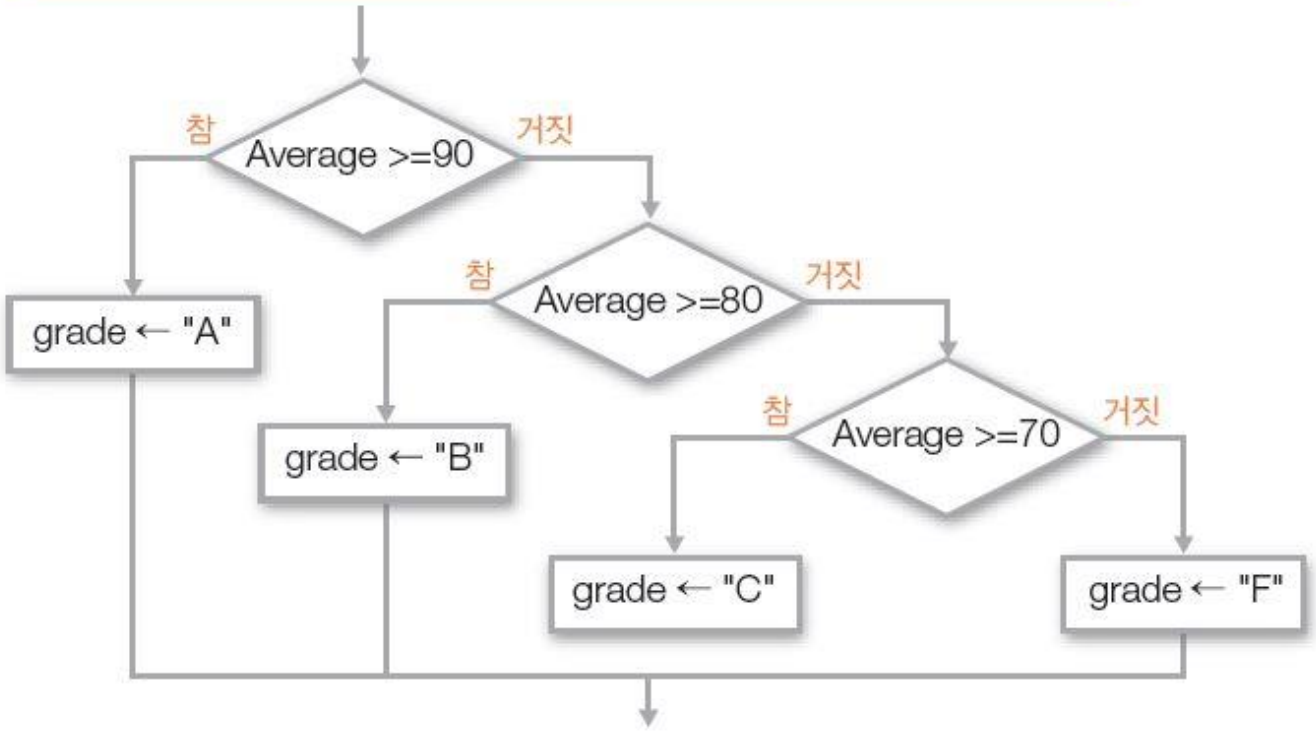


그림 1-31 중첩 if 문의 예

5. 알고리즘의 표현 방식

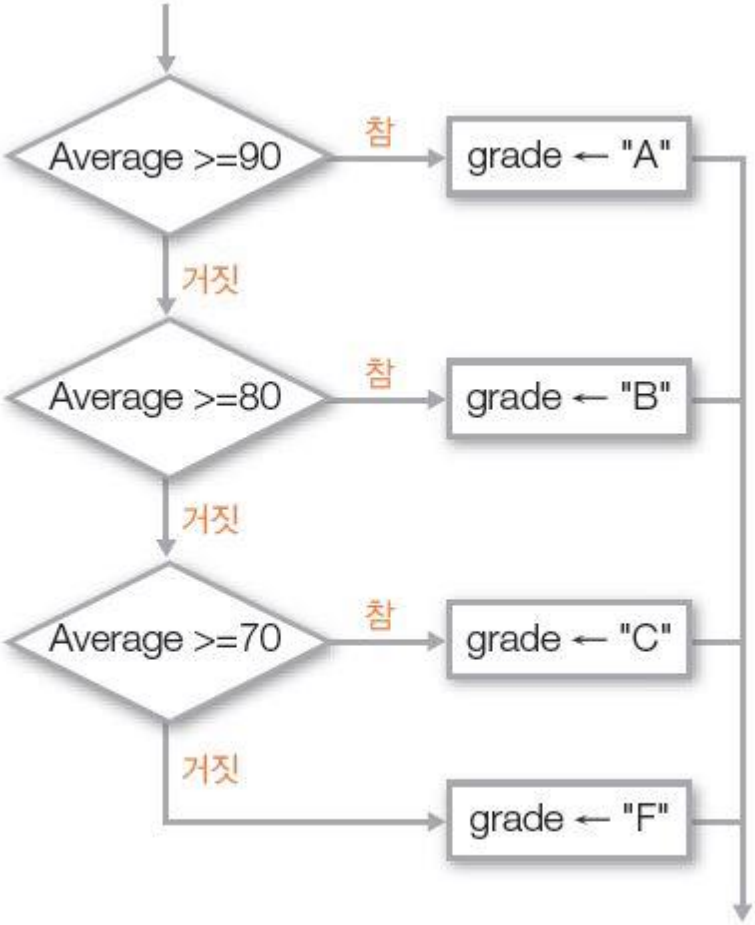
- case 문
  - 여러 조건식 중에서 해당 조건을 찾아서 그에 대한 명령문을 수행
  - 중첩 if 문으로 표현 가능
  - 형식과 제어흐름

```
case {
    조건식 1 : 명령문 1;
    조건식 2 : 명령문 2;
    ...
    조건식 n : 명령문 n;
    else    : 명령문 n+1;
}
```

그림 1-32 case 문의 형식과 제어 흐름

5. 알고리즘의 표현 방식

- case 문 예) 평균 점수에 따른 등급 계산하기



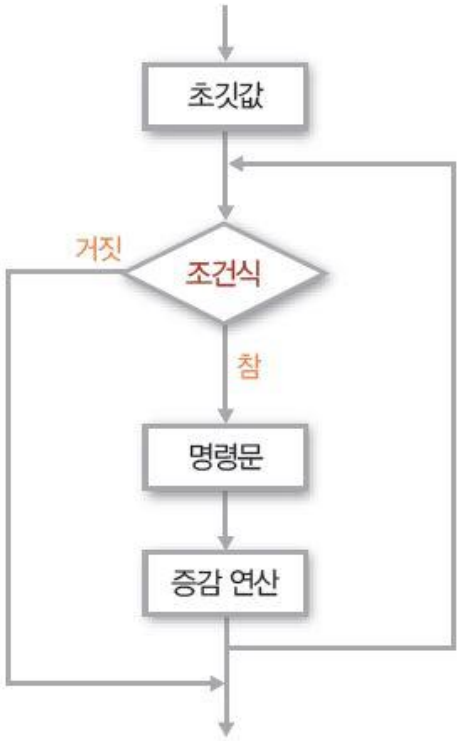

```
case {
    Average >= 90 : grade ← "A";
    Average >= 80 : grade ← "B";
    Average >= 70 : grade ← "C";
    else :          grade ← "F";
}
```

그림 1-33 case 문의 예

5. 알고리즘의 표현 방식

❖ 반복문

- 일정한 명령을 반복 수행하는 루프(loop) 형태의 제어구조
- for 문
  - 형식과 제어흐름

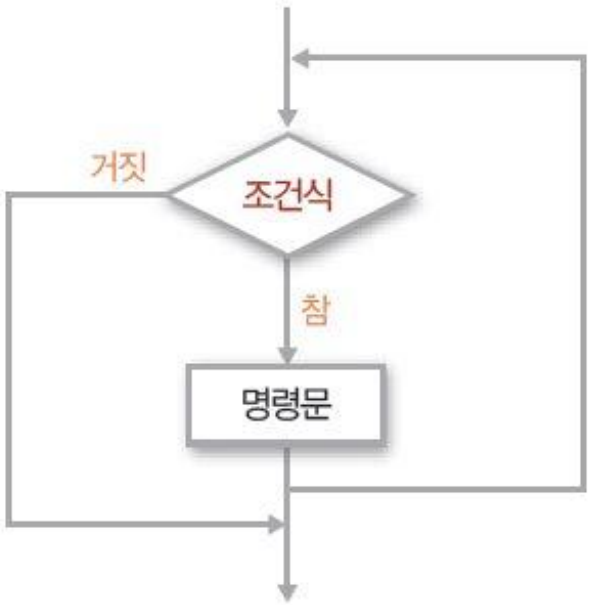



```
for (초깃값; 조건식; 증감값) do 명령문;
```

그림 1-34 for 문의 형식과 제어 흐름

5. 알고리즘의 표현 방식

- while – do 문
  - 형식과 제어흐름




```
while (조건식) do 명령문;
```

그림 1-35 while-do 문의 형식과 제어 흐름

5. 알고리즘의 표현 방식

- do-while 문
  - 형식과 제어흐름

```
do 명령문;
while (조건식);
```

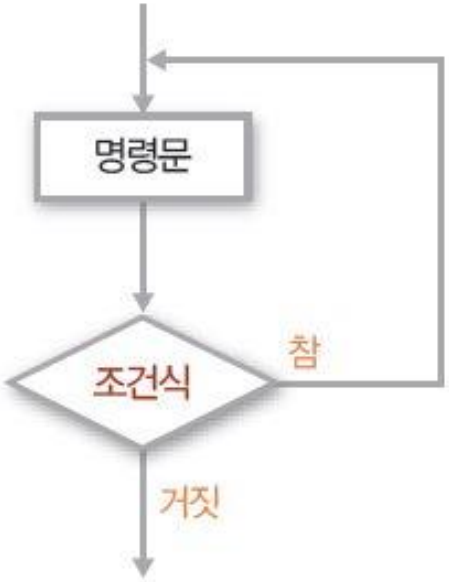


그림 1-36 do-while 문의 형식과 제어 흐름

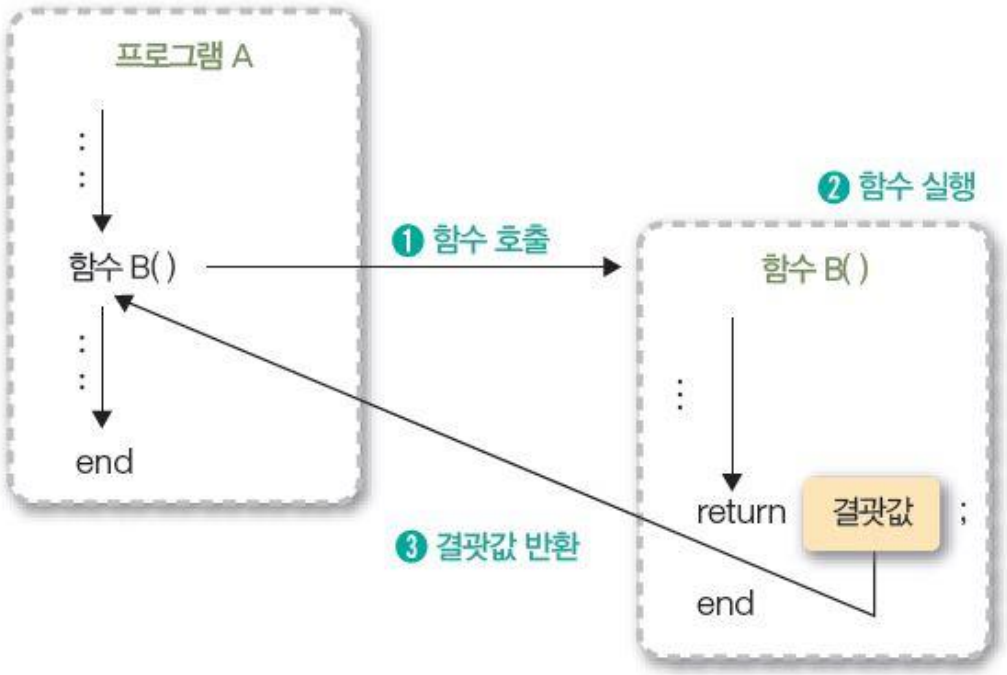
5. 알고리즘의 표현 방식

❖ 함수문

- 처리작업 별로 모듈화하여 만든 부프로그램
- 형식과 예

```
함수 이름 (매개변수)
    명령문;
    ...
    return 결괏값;
end
```

(a) 함수의 형식



(b) 함수의 호출과 실행 및 결괏값 반환의 예

그림 1-37 함수의 형식과 예

6. 알고리즘의 성능분석

❖ 알고리즘 성능 분석 기준

- 기준에는 정확성, 명확성, 수행량, 메모리 사용량, 최적성 등 있음
  - 정확성 : 올바른 자료 입력 시 유한한 시간 내에 올바른 결과 출력 여부
  - 명확성 : 알고리즘이 얼마나 이해하기 쉽고 명확하게 작성되었는가
  - 수행량 : 일반적인 연산 제외, 알고리즘 특성 나타내는 중요 연산 모두 분석
  - 메모리 사용량
  - 최적성 : 가장 중요

6. 알고리즘의 성능분석

❖ 알고리즘 성능 분석 방법

- 공간 복잡도
  - 알고리즘을 프로그램으로 실행하여 완료하기까지 필요한 총 저장 공간의 양
  - 공간 복잡도 = 고정 공간 + 가변 공간

- 시간 복잡도
  - 알고리즘을 프로그램으로 실행하여 완료하기까지의 총 소요시간
  - 시간 복잡도 = 컴파일 시간 + 실행 시간
    - 컴파일 시간 : 프로그램마다 거의 고정적인 시간 소요
    - 실행 시간 : 컴퓨터의 성능에 따라 달라질 수 있으므로 실제 실행시간 보다는 명령문의 실행 빈도수에 따라 계산
  - 실행 빈도수의 계산
    - 지정문, 조건문, 반복문 내의 제어문과 반환문은 실행시간 차이가 거의 없으므로 하나의 단위시간을 갖는 기본 명령문으로 취급

6. 알고리즘의 성능분석

❖ 알고리즘 성능 분석 표기법

- 빅-오 표기법
  - O(f(n))과 같이 표기, "Big Oh of f (n)"으로 읽음
  - 수학적 정의
    - 함수 f(n)과 g(n)이 주어졌을 때, 모든 n≥n0에 대하여 |f(n)| ≤ c |g(n)|을 만족하는 상수 c와 n0이 존재하면, f(n) = O(g(n))이다.
  - 함수의 상한을 나타내기 위한 표기법
    - 최악의 경우에도 g(n)의 수행 시간 안에는 알고리즘 수행 완료 보장
  - 먼저 실행 빈도수를 구하여 실행 시간 함수를 찾고, 이 함수값에 가장 큰 영향을 주는 n에 대한 항을 한 개 선택하여 계수는 생략하고 O의 오른쪽 괄호 안에 표시
  - [알고리즘 1-1]의 피보나치 수열에서 실행 시간 함수는 4n+2이고, 가장 영향이 큰 항은 4n인데 여기에서 계수 4를 생략하여 O(n)으로 표기

6. 알고리즘의 성능분석

- 각 실행 시간 함수에서 n값의 변화에 따른 실행 빈도수 비교

logn <	n <	nlogn <	n^2 <	n^3 <	2^n
0	1	0	1	1	2
1	2	2	4	8	4
2	4	8	16	64	16
3	8	24	64	512	256
4	16	64	256	4096	65536
5	32	160	1024	32768	4294967296

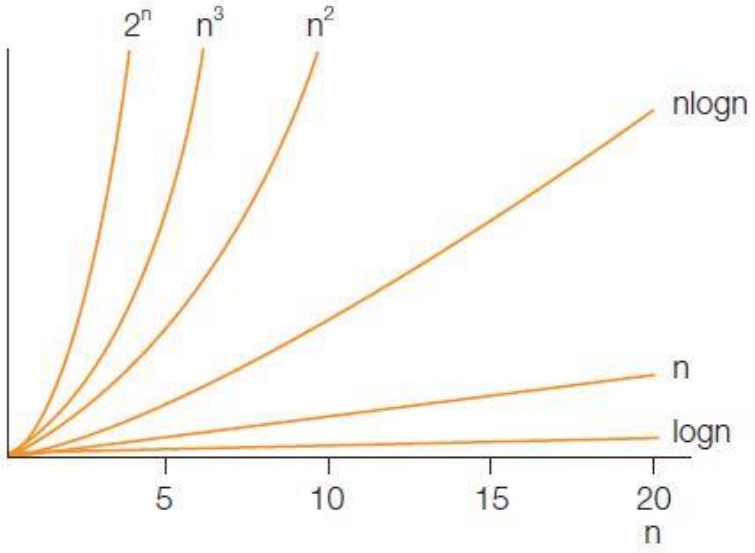


그림 1-38 실행 시간 함수에서 n값의 변화에 따른 실행 빈도수 비교

6. 알고리즘의 성능분석

- 시간 복잡도에 따른 알고리즘 수행 시간 비교

표 1-10 시간 복잡도에 따른 알고리즘 수행 시간 비교 예

입력 크기 n	알고리즘 수행 시간				
	n	nlogn	n^2	n^3	2^n
10	10^{-8}초	3×10^{-8}초	10^{-7}초	10^{-6}초	10^{-6}초
30	3×10^{-8}초	2×10^{-7}초	9×10^{-7}초	3×10^{-5}초	1초
50	5×10^{-8}초	3×10^{-7}초	3×10^{-6}초	10^{-4}초	13일
100	10^{-7}초	7×10^{-7}초	10^{-5}초	10^{-3}초	4×10^{13}년
1,000	10^{-6}초	10^{-5}초	10^{-3}초	1초	3×10^{283}년
10,000	10^{-5}초	10^{-4}초	10^{-1}초	17분	
100,000	10^{-4}초	2×10^{-3}초	10초	12일	
1,000,000	10^{-3}초	2×10^{-2}초	17분	32년	

Thank You