1 추상 자료형

1 뇌의 추상화 기능

기억할 대상의 구별되는 특징만을 단순화하여 기억하는 기능

※ 출처 : C로 배우는 쉬운 자료구조(개정3판), 이지영, 한빛미디어

길을 가다가
같은 과 친구인 철수를
보고 손을 흔들었다면

>> 우리의 뇌가 철수를 **기억(인식)**해낸 것

1 뇌의 추상화 기능

- 철수를 기억하기 위해 철수에 대한 모든 정보를 구체적이고 세세하게 저장하기는 어려움

  → 그 대신 철수를 다른 사람과 구별할 수 있는 철수만의 특징을 찾아서 기억함

> "자세하고 복잡한 것 대신 필수적이고 중요한 특징만 골라서 단순화시키는 작업이 **추상화** 작업임"

2 컴퓨터를 이용한 문재해결에서의 추상화

크고 복잡한 문제를 **단순화**시켜
쉽게 해결하기 위한 방법

2 컴퓨터를 이용한 문재해결에서의 추상화

자료 추상화(Data Abstraction)

: 처리할 자료, 연산, 자료형에 대한 추상화 표현

- 자료
  - 프로그램의 처리 대상이 되는 모든 것을 의미
  - 어떤값(Value)자체를 의미하기도 함

2 컴퓨터를 이용한 문재해결에서의 추상화

자료 추상화(Data Abstraction)

: 처리할 자료, 연산, 자료형에 대한 추상화 표현

- 연산
  - 어떤 일을 처리하는 과정으로 연산자를 사용하여 수행됨

    (ex. 더하기 연산은 +연산자에 의해 수행)

2 컴퓨터를 이용한 문재해결에서의 추상화

자료 추상화(Data Abstraction)

: 처리할 자료, 연산, 자료형에 대한 추상화 표현

- 자료형
  - 처리할 자료의 집합과 자료에 대해 수행할 연산자의 집합
  - 자료형을 정의할때는 자료형에 속하는 값과 이를 처리하기 위해 사용할 수 있는 연산자를 정의함

2 컴퓨터를 이용한 문재해결에서의 추상화

자료 추상화(Data Abstraction)

: 처리할 자료, 연산, 자료형에 대한 추상화 표현

- 자료형
  - 예를 들면 정수 자료형인 경우

자료	연산자
정수의 집합 {…, -1, 0, 1, …}	정수에 대한 연산자 집합 {+, -, x, ÷, mod}

3 추상 자료형(ADT, Abstract Data Type)

- 추상 자료형이란?

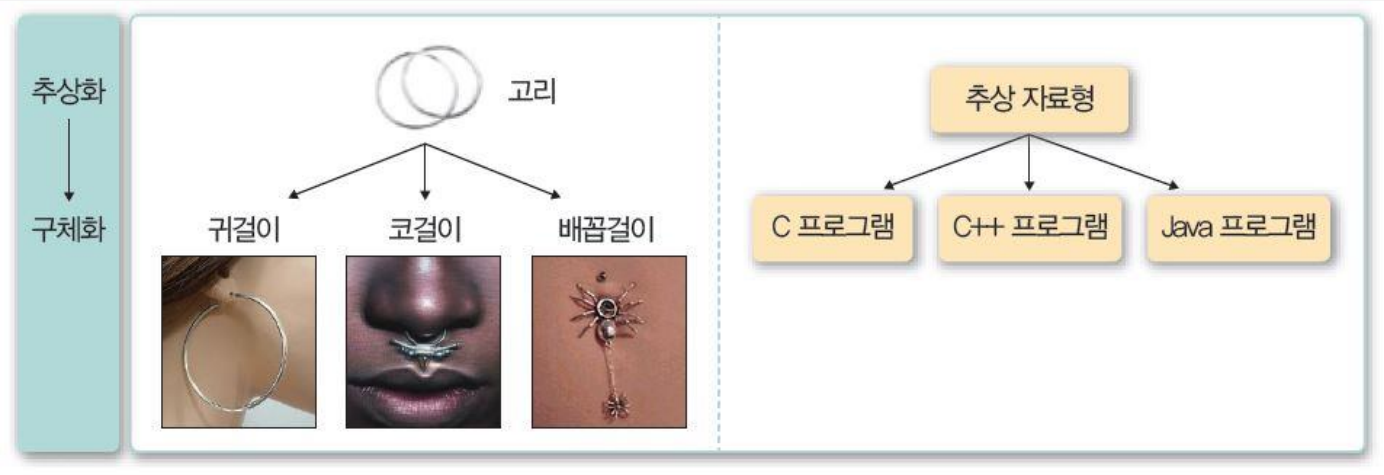
  자료형을 정의하려면 구체적으로 구현하기 전에 자료형에 대한 자료의 특성, 연산자, 연산자가 무엇을 수행하는지 등 자료와 연산자의 특성을 논리적으로 추상화하여 정의한 자료형

3 추상 자료형(ADT, Abstract Data Type)

- 추상화와 구체화
  - 추상화 - **"무엇(What)인가?"**를 논리적으로 정의
  - 구체화 - **"어떻게(How) 할 것인가?"**를 실제적으로 표현

3 추상 자료형(ADT, Abstract Data Type)

- 같은 사물이라도 **"어떻게(How) 쓰느냐"**에 따라 용도가 달라짐



※ 출처 : C로 배우는 쉬운 자료구조(개정3판), 이지영, 한빛미디어

- 기본틀 : '고리'는 추상화된 기본틀
- 구체화 : 기본틀을 용도에 맞게 귀걸이나 코걸이로 사용하는 것

3 추상 자료형(ADT, Abstract Data Type)

- 자료와 연산에 있어서의 추상화와 구체화의 관계

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

2 알고리즘 개념

1 알고리즘이란?

주어진 문제해결 방법을 추상화하여
단계적 절차를 논리적으로 기술해 놓은 명세서

- 효과적이고 정확하게 문제를 해결하려면 자료를 정의하고 그 자료를 철하기에 적당한 알고리즘을 작성해야 함

1 알고리즘이란?

알고리즘의 조건

- 입력(Input)
  : 알고리즘 수행에 필요한 자료가 외부에서
  입력으로 제공될 수 있어야 함

- 출력(Output)
  : 알고리즘 수행 후 하나 이상의 결과를
  출력해야 함

- 명확성(Definiteness)
  : 수행할 작업의 내용과 순서를 나타내는
  알고리즘의 명령어들은 명확하게 명세되어야 함

1 알고리즘이란?

알고리즘의 조건

- 유한성(Finiteness)
  : 알고리즘은 수행 뒤에 반드시 종료되어야 함

- 효과성(Effectiveness)
  : 알고리즘의 모든 명령어들은 기본적이며
  실행이 가능해야 함

1 알고리즘이란?

예시 딸기 시럽을 얹은 치즈 케이크 만드는 레시피의 알고리즘과 자료의 예

요리 재료
= **컴퓨터의 자료**

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

알고리즘

요리법

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. ②에 레몬즙과 바닐라 에센스를 넣고 살짝 저은 다음 ①에 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 ④를 뿌린 다음 플레인 요구르트와 딸기를 얹는다.

연산

케익을 만드는 과정을 기술한 요리법
= **알고리즘**

요리 재료를 다루는 방법
= **자료에 대한 연산**

※ 출처 : C언어로 쉽게 풀어쓴 자료구조 (개정판), 천인국외 2명, 생능출판사

2 알고리즘의 표현 방법의 종류

자연어를 이용한 서술적 표현 방법

: 알고리즘을 사람이 쓰는 자연어로 표현하는 방법

- 자연어는 서술적일 뿐만 아니라 쓰는 사람에 따라 일관성이나 명확성을 유지하기 어려움
- 따라서 누구라도 쉽게 이해하고 쓸 수 있어야 하는 알고리즘을 표현하는데 한계가 있음

2 알고리즘의 표현 방법의 종류

순서도(Flow chart)를 이용한 도식화 표현 방법

: 알고리즘을 순서도를 작성하는 규칙에 따라 도식화하는 방법

- 순서도를 이용하면 명령의 흐름을 쉽게 파악할 수 있지만 복잡한 알고리즘을 표현하는 데는 한계가 있음

2 알고리즘의 표현 방법의 종류

프로그래밍 언어를 이용한 구체화 방법

: 알고리즘을 프로그램언어를 사용하여 표현하는 방법

- 이 방법을 사용하면 알고리즘 자체가 구체화되므로 추가로 구체화 작업을 할 필요가 없음
- 하지만 특정한 프로그래밍 언어로 작성하기 때문에 해당 언어를 모르면 이해하기 어려움
- 다른 프로그래밍 언어로 프로그램을 개발하는 경우에는 알고리즘을 번역하고 다른 프로그래밍 언어로 변환해야 하므로 비효율적임

2 알고리즘의 표현 방법의 종류

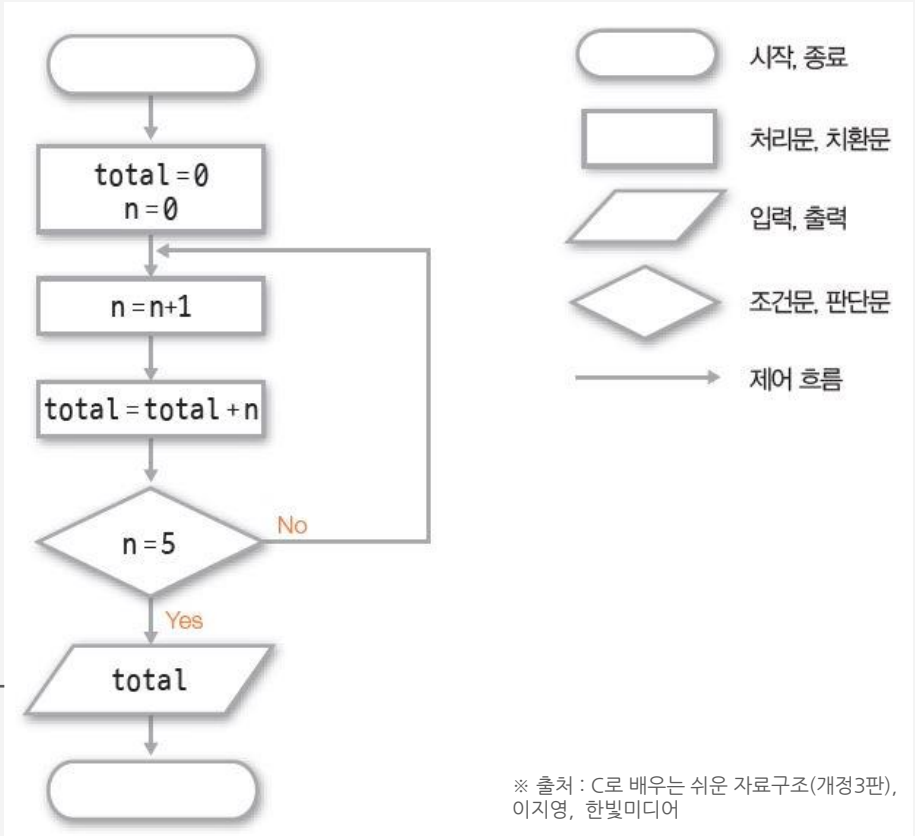
가상코드(Pseudo-code)를 이용한 추상화 방법

: 알고리즘을 프로그래밍 언어로 표현했을 때 생기는 단점을 보완하는 방법

- 특정 프로그래밍 언어는 아니지만 프로그래밍 언어의 형태를 갖춘 가상코드를 사용하여 알고리즘을 표현함
- 가상코드는 프로그래밍 언어가 아니므로 직접 실행할 수는 없지만, 형태가 일반적인 프로그래밍 언어와 유사하기 때문에 원하는 특정 프로그래밍 언어로 변화(구체화 작업)하기가 쉬움

3 순서도를 이용한 도식화

- 순서도는 특정 기호를 사용하여 알고리즘의 실행 단계를 표현함



※ 출처 : C로 배우는 쉬운 자료구조(개정3판), 이지영, 한빛미디어

순서도의 예시 >>
1부터 5까지의 합을 구하는 알고리즘

4 가상코드를 이용한 추상화

- 가상코드는 의사코드 또는 알고리즘 기술언어 (ADL, Algorithm Description Language)로도 부름
- 의사코드를 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 가상코드의 형식의 기본 요소

기호

- 변수, 자료형 이름, 프로그램 이름, 레코드 필드 명, 문장의 레이블 등을 나타냄
- 문자나 숫자를 조합한 것으로 첫 글자는 반드시 영문자로 사용
- 문장의 레이블 다음에는 콜론(:)을 입력해 수행할 문장과 구분함

5 가상코드의 형식의 기본 요소

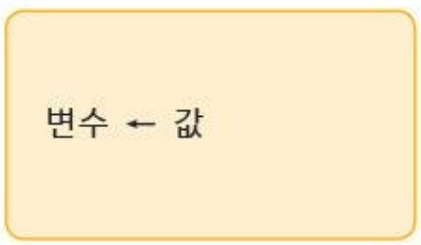
자료형

- 정수형과 실수형의 수치 자료형, 문자형, 논리형, 포인터, 문자열 등의 모든 자료형을 사용

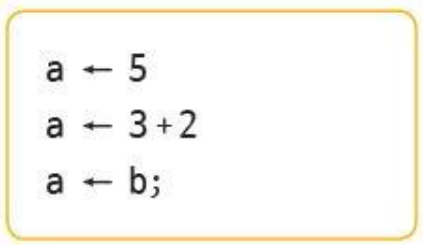
5 가상코드의 형식의 기본 요소

연산자

- 산술연산자, 관계연산자, 논리연산자



(a) 지정문 형식

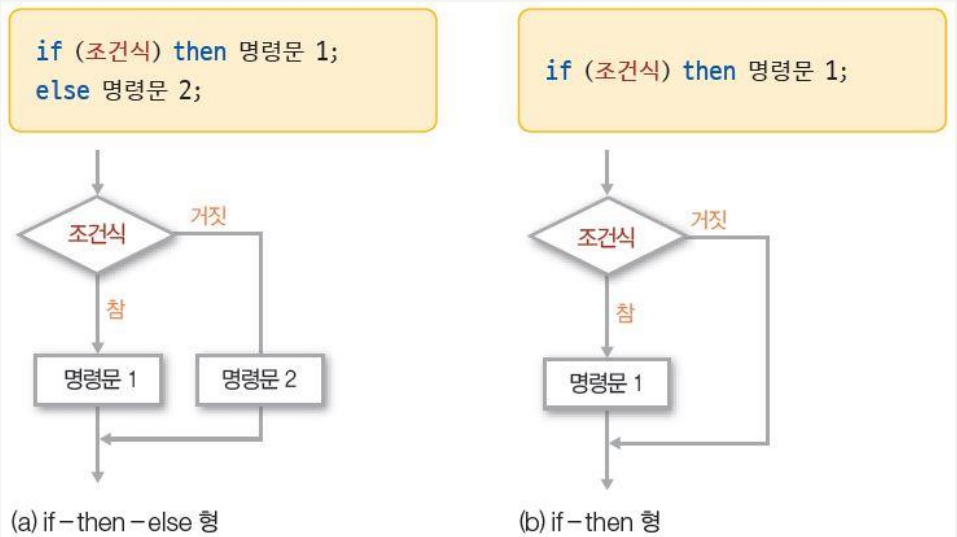


(b) 지정문 예

※ 출처 : C로 배우는 쉬운 자료구조(개정3판), 이지영, 한빛미디어

5 가상코드의 형식의 기본 요소 : 조건문

- 조건에 따라 실행할 명령문이 결정되는 선택적 제어구조를 만듦



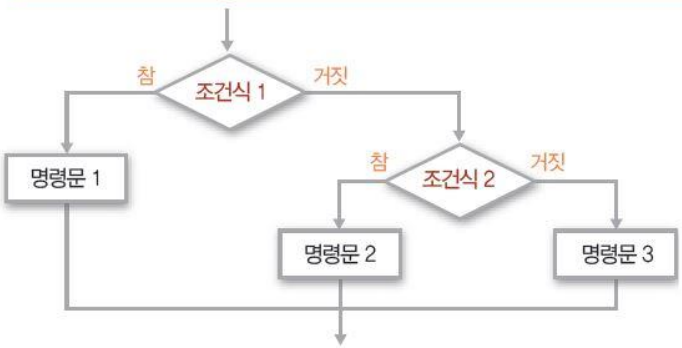
※ 출처 : C로 배우는 쉬운 자료구조(개정3판), 이지영, 한빛미디어

〈if 문의 형식과 제어흐름〉

5 가상코드의 형식의 기본 요소 : 다단계 조건문

- if문을 중첩해서 사용할 수 있음

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



※ 출처 : C로 배우는 쉬운 자료구조(개정3판), 이지영, 한빛미디어

〈중첩 if 문의 형식과 제어 흐름〉

5 가상코드의 형식의 기본 요소 : case 문

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

5 가상코드의 형식의 기본 요소 : case 문

case문의 형식과 제어흐름

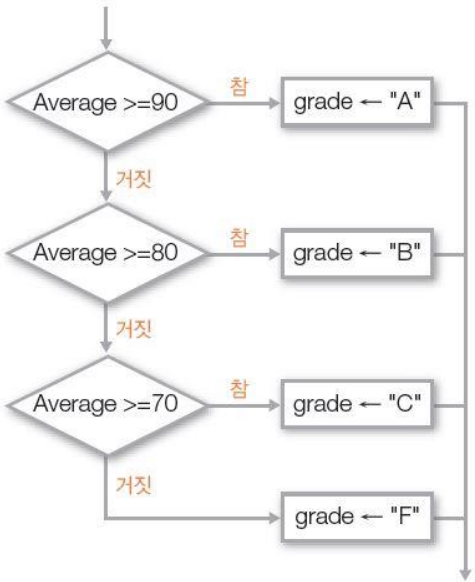
```
case {
    조건식 1 : 명령문 1;
    조건식 2 : 명령문 2;
    ...
    조건식 n : 명령문 n;
    else    : 명령문 n+1;
}
```

※ 출처 : C로 배우는 쉬운 자료구조(개정3판), 이지영, 한빛미디어

5 가상코드의 형식의 기본 요소 : case 문

case 문의 예시

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



※ 출처 : C로 배우는 쉬운 자료구조(개정3판), 이지영, 한빛미디어

5 가상코드의 형식의 기본 요소 : 반복문

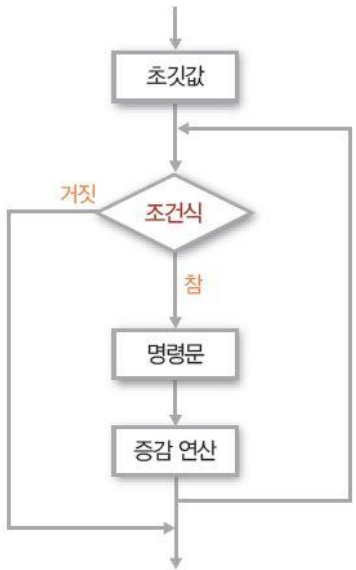
- 일정한 명령을 반복 수행하는 루프(loop) 형태의 제어구조

반복문의 종류

: for, while-do, do-while 문

5 가상코드의 형식의 기본 요소 : for문

for 문의 형식과 제어흐름



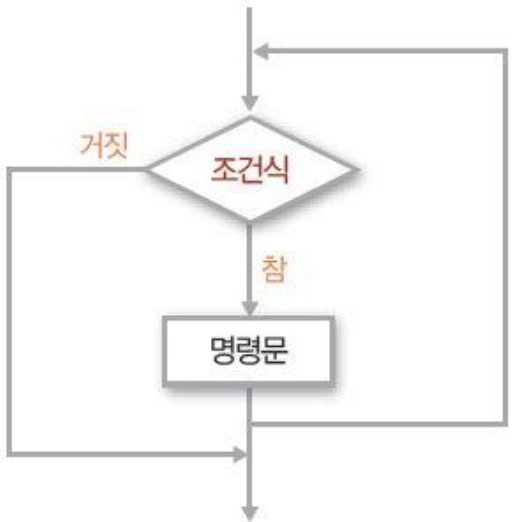

```
for (초깃값; 조건식; 증감값) do 명령문;
```

※ 출처 : C로 배우는 쉬운 자료구조(개정3판), 이지영, 한빛미디어

5 가상코드의 형식의 기본 요소 : while - do 문

- 조건식을 검사하여 조건식이 참인 동안 명령문을 반복하여 수행함

```
while (조건식) do 명령문;
```

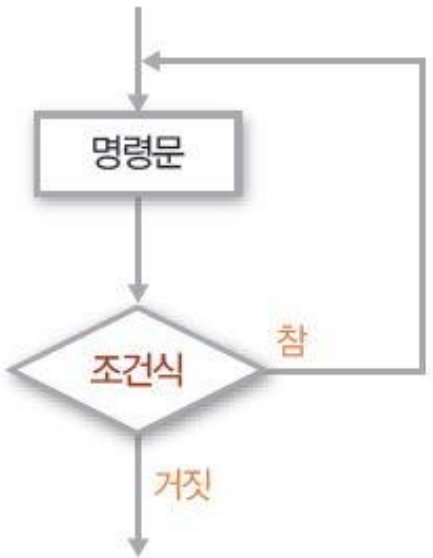


※ 출처 : C로 배우는 쉬운 자료구조(개정3판), 이지영, 한빛미디어

5 가상코드의 형식의 기본 요소 : do - while 문

- 조건이 참이 아니더라고 명령문이 한번은 수행됨

```
do 명령문;
while (조건식);
```



※ 출처 : C로 배우는 쉬운 자료구조(개정3판), 이지영, 한빛미디어

5 가상코드의 형식의 기본 요소 : 함수문

처리작업 별로 모듈화하여 만든 부프로그램

- 어떤 문제를 처리하는 프로그램을 만들 때 프로그램을 한 개로 구성하는 것 보다 처리할 작업별로 모듈화하여 작은 단위프로그램을 여러 개 구성하는 것이 좋을 수도 있음

- 전체 프로그램 중 같은 작업은 하나의 단위 프로그램에서 독립적으로 수행하게 되면 프로그램 크기가 줄어들고 수정과 관리가 쉬울 뿐만 아니라 다른 프로그램에서도 단위 프로그램을 재사용할 수 있음

5 가상코드의 형식의 기본 요소 : 함수문

처리작업 별로 모듈화하여 만든 부프로그램

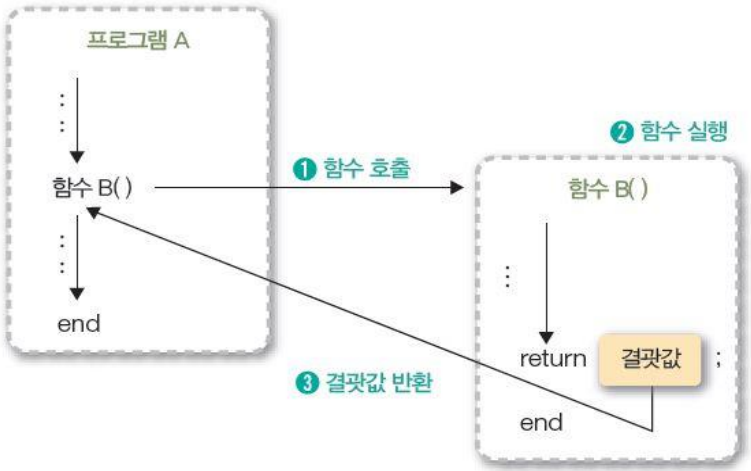
- 함수는 함수를 호출했을 때 독립적으로 실행되며 매개변수를 사용하여 다른 함수나 프로그램을 서로 연결함

5 가상코드의 형식의 기본 요소 : 함수문

- return 문은 함수의 실행 결과값을 함수를 호출한 위치로 반환함 그리고 End문은 함수 실행을 마침

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

(a) 함수의 형식



(b) 함수의 호출과 실행 및 결괏값 반환의 예

※ 출처 : C로 배우는 쉬운 자료구조(개정3판), 이지영, 한빛미디어

3 알고리즘 성능분석 방법

3 | 알고리즘 성능분석 방법

1 알고리즘 성능 분석 기준

- 기준에는 정확성, 명확성, 수행량, 메모리 사용량, 최적성 등 있음

정확성	명확성	수행량	메모리 사용량	최적성

3 | 알고리즘 성능분석 방법

1 알고리즘 성능 분석 기준

정확성

: 올바른 자료 입력 시 유한한 시간 내에 올바른 결과 출력 여부

명확성

: 알고리즘이 얼마나 이해하기 쉽고 명확하게 작성되었는냐를 말함

수행량

: 기본적으로 포함되는 일반적인 연산은 제외하고 알고리즘 특성 나타내는 중요 연산을 모두 분석 해야 함

3 | 알고리즘 성능분석 방법

1 알고리즘 성능 분석 기준

메모리 사용량

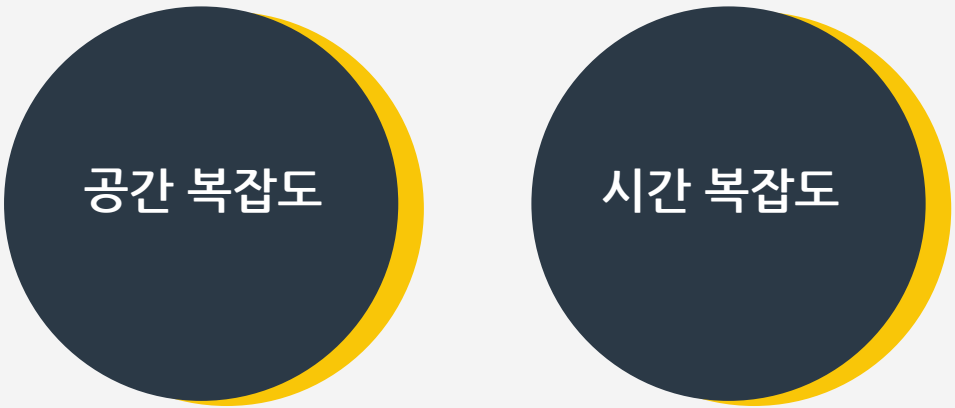
: 사용하는 명령어, 변수, 입출력 자료와 정보를 저장하기 위해 메모리 사용

최적성

: 가장 중요, 가장 좋은 알고리즘은 조건에 맞는 “최적의”알고리즘이라고 할 수 있음

2 알고리즘 성능 분석 방법

- 일반적으로 실행에 필요한 공간 측면에서 분석하는 **공간 복잡도**와 실행에 소요되는 시간측면에서 분석하는 **시간 복잡도**를 추정하여 평가함



2 알고리즘 성능 분석 방법

공간 복잡도

알고리즘을 프로그램으로 실행하여
완료하기까지 필요한 총 저장 공간의 양

공간 복잡도 = 고정 공간 + 가변 공간

2 알고리즘 성능 분석 방법

공간 복잡도

- 고정공간
  : 프로그램 크기나 입출력 회수와 상관없이 고정적으로 필요한 저장공간으로 프로그램 저장 공간과 변수 및 상수를 저장하는 공간

- 가변공간
  : 실행과정에서 사용되는 자료와 변수를 저장하는 공간, 함수를 실행하는데 관련 있는 정보를 저장하는 공간

2 알고리즘 성능 분석 방법

시간 복잡도

알고리즘을 프로그램으로 실행하여
완료하기까지의 총 소요시간

시간 복잡도 = 컴파일 시간 + 실행 시간

2 알고리즘 성능 분석 방법

시간 복잡도

- 컴파일 시간
  : 프로그램마다 거의 고정적인 시간 소요

- 실행 시간
  : 컴퓨터의 성능에 따라 달라질 수 있으므로
  실제 실행시간 보다는 명령문의 실행 빈도수에
  따라 계산

2 알고리즘 성능 분석 방법

시간 복잡도

- 실행 빈도수의 계산
  : 지정문, 조건문, 반복문 내의 제어문과
  반환문은 실행시간 차이가 거의 없으므로
  하나의 단위시간을 갖는 기본 명령문으로 취급

3 알고리즘 성능 분석 표기법

빅-오 표기법

- O(f(n))과 같이 표기, “Big Oh of f (n)”으로 읽음
- 함수의 상한을 나타내기 위한 표기법

  : 최악의 경우에도 수행 시간 안에는
  알고리즘 수행 완료 보장

- 먼저 실행 빈도수를 구하여 실행 시간 함수 f(n)을 찾고, 이 함수값에 가장 큰 영향을 주는 n에 대한 항을 한 개 선택하여 계수는 생략하고 O의 오른쪽 괄호 안에 표시

3 알고리즘 성능 분석 표기법

빅-오메가 표기법

: Ω(f(n))과 같이 표기, “Big Omega of f (n)”으로 읽음

- 함수의 하한을 나타내기 위한 표기법

  : 어떤 알고리즘의 시간 복잡도가 Ω(f(n))으로 분석되었다면, 이 알고리즘 수행에는 적어도 f(n)의 수행 시간이 필요함을 의미

3 알고리즘 성능 분석 표기법

빅-세타 표기법

: θ(f(n))과 같이 표기, “Big Theta of f (n)”으로 읽음

- 상한과 하한이 같은 정확한 차수를 표현하기 위한 표기법

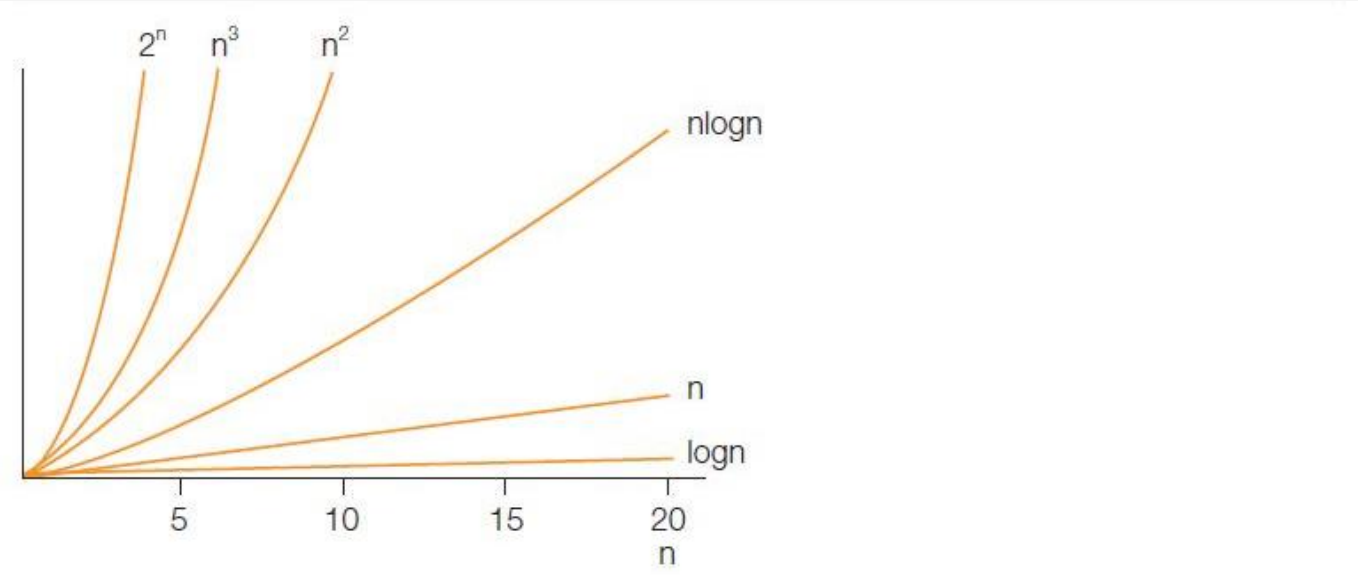
  : f (n)= θ (g(n))이 되려면 f (n)= O(g(n))이면서
  f (n)= Ω(g(n))이어야 함

3 알고리즘 성능 분석 표기법

- 시간복잡도를 정확히 계산할 수 있다면 **'빅-세타 표기법'**을 사용함
- 시간 복잡도를 정확히 분석하기 어렵다면 상한을 구하여 **'빅-오 표기법'**으로 사용하거나 하한을 구하여 **'빅-오메가 표기법'**을 사용함
- 일반적으로 최악의 경우를 고려한 해결책을 찾기 때문에 **'빅-오 표기법'**을 주로 사용함

3 알고리즘 성능 분석 표기법

- 각 실행 시간 함수에서 n값의 변화에 따른 실행 빈도수 비교



※ 출처 : C로 배우는 쉬운 자료구조 (개정3판), 이지영, 한빛미디어

3 알고리즘 성능 분석 표기법

시간 복잡도에 따른 알고리즘 수행 시간 비교

※ 출처 : C로 배우는 쉬운 자료구조(개정3판), 이지영, 한빛미디어

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	