1 알고리즘의 정의

1 알고리즘의 정의

1 알고리즘(Algorithm)

- 문제를 해결하기 위한 체계적 단계
- 어떤 문제에 대해 입력을 받아서 출력을 내기 위해서 컴퓨터에 의해 실행 가능하고 유한 번 수행 후에 종료되는 명확한 명령어들의 나열
- 문제를 해결하기 위한 명령어들의 유한 집합
- 알고리즘을 설계하기 위해서는 해야 할 작업을 명확하게 명시해야 함
- 문제 해결이나 처리 과정에서의 순서를 단계적으로 서술

1 알고리즘의 정의

1 알고리즘(Algorithm)

◆ 예) 일반적인 언어의 알고리즘으로
지하철 타는 방법을 작성하시오

(풀이)
① 지하철역으로 간다.
② 티켓 자판기로 간다.
③ 자판기에 출발역, 도착역을 입력한다.
④ 해당 금액을 투입한다.
⑤ 티켓을 받는다.
⑥ 개찰구로 간다.
⑦ 개찰구에 표를 댄다.
⑧ 개찰구 문이 열리면 들어간다.
⑨ 지하철이 올 때까지 기다린다.
⑩ 지하철이 오면 객차에 탄다.

1 알고리즘의 정의

2 알고리즘의 조건

- 입력(Input)
  - 문제와 관련된 입력이 반드시 존재해야 함
- 출력(output)
  - 문제를 해결했을 때
    그 결과인 출력이 반드시 존재해야 함
- 유한성(Finiteness)
  - 입력은 제한된 개수의 명령 단계를 거쳐
    출력을 내고 반드시 종료되어야 함
  - 알고리즘 수행이 끝나지 않거나
    매우 오래 걸리면 해를 얻을 수 없음

1 알고리즘의 정의

2 알고리즘의 조건

- 정확성(Correctness)
  - 입력을 이용한 문제 해결 과정과 출력은 논리적이고 정확해야 함
- 일반성(Generality)
  - 같은 유형의 문제에 항상 적용될 수 있어야 함

1 알고리즘의 정의

3 알고리즘의 표현 방법

- 순서도(Flow Chart)
  - 문제 해결 과정을
    기호와 도형을 사용하여 표현하는 방식

- 의사코드(Pseudocode)
  - 프로그램 명령문 형식을 취하고
    각 명령을 사람이 이해하기 쉽게
    적당한 뜻을 가진 단어로 나타냄
  - 특정 프로그래밍 언어가 아니므로
    직접 실행은 불가능
  - 일반적인 프로그래밍 언어의 형태이므로
    원하는 특정 프로그래밍 언어로의 변환 용이

1 알고리즘의 정의

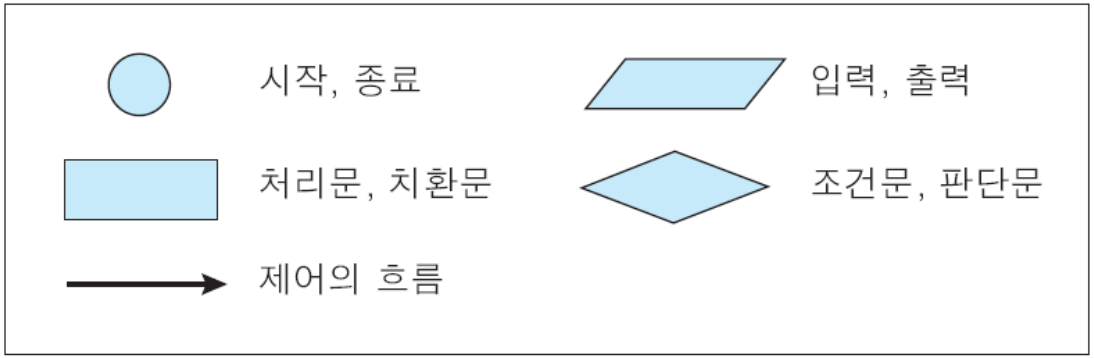
3 알고리즘의 표현 방법

- 일반적인 언어
  - 사람이 사용하는 문장으로 설명

➡ 알고리즘은 여러 가지 방법으로 표현 가능

4 순서도를 이용한 알고리즘의 표현

- 순서도에서 사용하는 기호

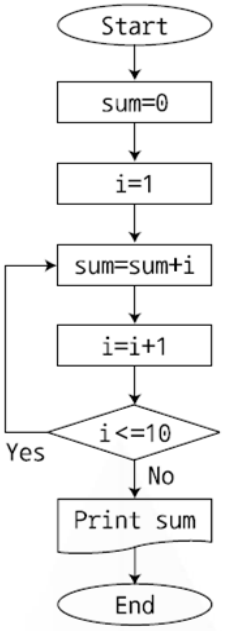


- 장점 : 알고리즘의 흐름 파악이 용이함
- 단점 : 복잡한 알고리즘의 표현이 어려움

1 알고리즘의 정의

4 순서도를 이용한 알고리즘의 표현

- 예) 1부터 10까지의 합을 구하는 과정을 순서도로 표현



순서도

※출처: 컴퓨팅 사고력을 키우는 이산수학, 박주미, 한빛아카데미

5 의사코드의 기본 요소

- 기호
  - 변수, 자료형 이름, 프로그램 이름,
    레코드 필드명, 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

6 의사코드의 형식

- 대입문
  - 사용 형식 `변수 ← 값;`
    - 대입연산자(←)의 오른쪽에 있는 값
      (식의 계산 결과값이나 변수값)을
      대입연산자(←)의 왼쪽에 있는 변수에 저장

6 의사코드의 형식

- 조건문
  - 조건에 따라 실행할 명령문이 결정되는 선택적 제어 구조
  - if 문, case 문
    - if문 형식

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

또는

```
if (조건식) then 명령문 1;
```

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

6 의사코드의 형식

- 반복문
  - 일정한 명령을 반복 수행하는 루프(loop) 형태의 제어 구조
  - for 문, while 문, do~while 문
    - for 문 형식
      - 형식 : for (초기값 ; 조건식 ; 증감값) do 명령문 ;
      - 초기값 : 반복문을 시작하는 값
      - 조건식 : 반복 수행 여부를 검사하는 조건식
      - 증감값 : 반복 회수를 계산하기 위해서 반복문을 한번 수행할 때마다 증가 또는 감소시키는 값

7 의사코드를 이용한 알고리즘의 표현

◆ 예) 변수 a, b, c를 입력받아 가장 큰 수를 판별하는 프로그램에 대한 알고리즘을 의사코드로 작성하라

(풀이)

```
algorithm maxnum(a, b, c){
    x = a;
    if b > x then
        x = b;
    else if c > x then
        x = c;
    endif
    print x;
}
```

7 의사코드를 이용한 알고리즘의 표현

◆ 예) 다음 알고리즘의 결과는 무엇인가?

```
algorithm B( ){
    i = 1;
    x = 0;
    while(i<=n)
        x=x+1;
        i=i+1;
    endwhile
    printf i, x;
}
```

(풀이)
n 값을 알 수 없기 때문에 반복되지 않거나 무한 반복, 이 알고리즘은 유한적이고 논리적인 명령을 실행할 수 없음.

➡ 따라서 정확한 출력을 얻을 수 없는 알고리즘임

2 기본 자료구조

2 기본 자료구조

1 자료구조(Data Structure)

- 컴퓨터가 효율적으로 문제를 처리하기 위해서는 문제를 정의하고 분석하여 그에 대한 최적의 프로그램을 작성해야 함

- 컴퓨터를 이용해 문제를 해결하려면 우선 해당 문제를 컴퓨터가 처리할 수 있는 형태(자료구조)로 표현해야 함

- 이때 자주 사용되는 자료구조는 기본 자료형, 배열, 연결 리스트, 큐, 스택, 트리, 그래프 등이 있음

2 알고리즘과의 관계

알고리즘 + 자료구조 = 프로그램

2 기본 자료구조

3 배열

- 동일한 자료형의 데이터를 하나의 이름으로 모아 놓은 데이터의 집합
- 각각의 데이터는 번호(인덱스)를 통해 구분
- 배열의 데이터는 연속된 메모리 공간에 순차적으로 저장됨

3 배열

장점과 단점

장점	변수의 시작 위치와 하나의 데이터의 크기만 알면 배열의 원소에 바로 접근할 수 있음
단점	데이터는 항상 순차적으로 저장되어야 하므로 원소에 대한 삽입, 삭제가 있을 경우에는 기존의 데이터를 이동시키거나 복사해야 함 ➔ 이러한 배열의 문제점을 보완한 자료구조가 연결 리스트임

4 연결 리스트

- 연결 리스트는 각 노드에 그 다음 노드가 어디에 저장되어 있는지를 가리키는 주소가 함께 저장되기 때문에 기억장치 내의 어느 위치에든 자료를 저장할 수 있음

단점	
	• 특정 데이터에 바로 접근할 수 없음 • Head부터 시작하여 해당 노드가 가리키고 있는 주소의 노드를 따라가면 원하는 데이터를 찾음 • 데이터 개수가 많고 임의의 원소를 읽는 경우가 빈번하다면 연결 리스트는 효과적이지 않음

5 자료의 형태에 따른 분류

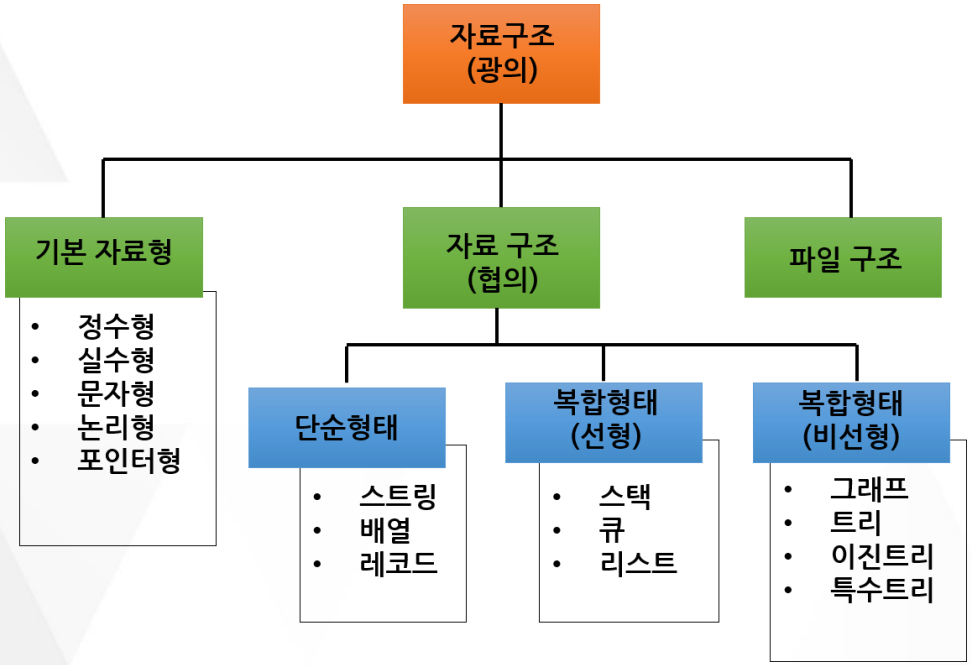
- 단순 구조
  - 정수, 실수, 문자, 문자열 등의 기본 자료형
- 선형 구조
  - 자료들 사이의 관계가 1:1 관계
  - 순차 리스트, 연결 리스트, 스택, 큐, 데크 등
- 비선형 구조
  - 자료들 사이의 관계가 1:다, 또는 다:다 관계
  - 트리, 그래프 등

5 자료의 형태에 따른 분류

- 파일 구조
  - 서로 관련 있는 필드로 구성된 레코드의 집합인 파일에 대한 구조
  - 순차 파일, 색인 파일, 직접 파일 등

2 기본 자료구조

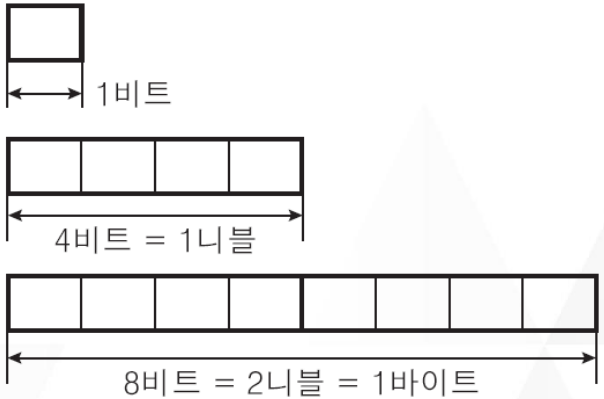
6 자료구조의 분류



※출처: 알기쉽게 해설한 이산수학, 손진곤, 이한미디어

7 컴퓨터에서의 자료 표현

- 숫자, 문자, 그림, 소리, 기호 등 모든 형식의 자료를 2진수 코드로 표현하여 저장 및 처리
- 2진수 코드
  - 1과 0, ON과 OFF, 참(True)과 거짓(False)의 조합
- 2진수 코드의 단위



※출처: 자바로 배우는 쉬운 자료구조, 이지영, 한빛아카데미

3 알고리즘의 효율성

3 알고리즘의 효율성

1 바람직한 알고리즘

- 명확해야 함
  - 이해하기 쉽고 가능하면 간명하도록 해야 함
  - 지나친 기호적 표현은 오히려 명확성을 떨어뜨림
  - 단순한 알고리즘일수록 인간이 이해하기 쉽고 다른 알고리즘으로 변형하기도 좋고 성능이 좋은 경우도 많음
  - 명확성을 해치지 않으면 일반 언어의 사용도 무방

- 효율적이어야 함
  - 같은 문제를 해결하는 알고리즘들의 수행 시간이 수백만 배 이상 차이가 날 수 있음

3 알고리즘의 효율성

2 알고리즘 분석의 필요성

- 하나의 문제는 다양한 방법의
  알고리즘을 통해 해결될 수 있으며
  각각의 알고리즘 성능을 평가하는 것은 중요한 문제임

- 바람직한 알고리즘은 명확하고 효율적이어야 함

- 알고리즘을 설계하고 나면 이 알고리즘이
  자원을 얼마나 소모하는지 분석해야 함

3 알고리즘의 효율성

2 알고리즘 분석의 필요성

- 자원은 소요 시간, 메모리, 통신 대역 등
- 자원의 가장 중심 대상은 소요시간
- 시간의 분석은 최악의 경우과
  평균적인 경우에 대한 분석이 대표적

  ➡ 시간 분석을 하면 알고리즘이 어느 정도의
  입력에서 어느 정도의 시간이 필요한지
  미리 짐작 가능하며 주어진 시간에 요구하는
  작업이 완료 가능한지를 알 수 있음

3 알고리즘의 효율성

3 알고리즘 분석의 기준

- 정확성(Correctness)
  - 올바른 입력에 대하여 유한 시간 내에 올바른 출력을 제공하는 것
- 수행량(Amount of work done)
  - 알고리즘에 의한 수행되는 일의 양
  - 기본 연산의 처리량
  - 컴퓨터, 프로그래밍 언어, 프로그래머 등과 독립적인 수행량 평가 기준이 필요
- 기억장소 사용량(Amount of space used)
  - 알고리즘 수행에 따라 요구되는 기억 장소의 양
  - 알고리즘을 어떻게 구현하였는가에 따라 달라짐

3 알고리즘의 효율성

3 알고리즘 분석의 기준

- 단순성(Simplicity)
  - 알고리즘이 단순
    - ⋯› 알고리즘의 정확성을 쉽게 증명
    - ⋯› 프로그램 작성이 쉬움
    - ⋯› 프로그램의 수정도 용이해짐

- 최적성(Optimality)
  - "알고리즘이 최적"
    - ⋯› 해당 알고리즘보다 더 적은 기본 연산을 수행하는 알고리즘은 없음