



Part 01

알고리즘

1. 알고리즘 이해

- 알고리즘과 순서도, 순서도의 기본 개념,
순서도 작성법, 변수와 배열, 연산과 함수,
알고리즘 검증

2. 기본 알고리즘 - 수열

3. 기본 알고리즘 - 수학

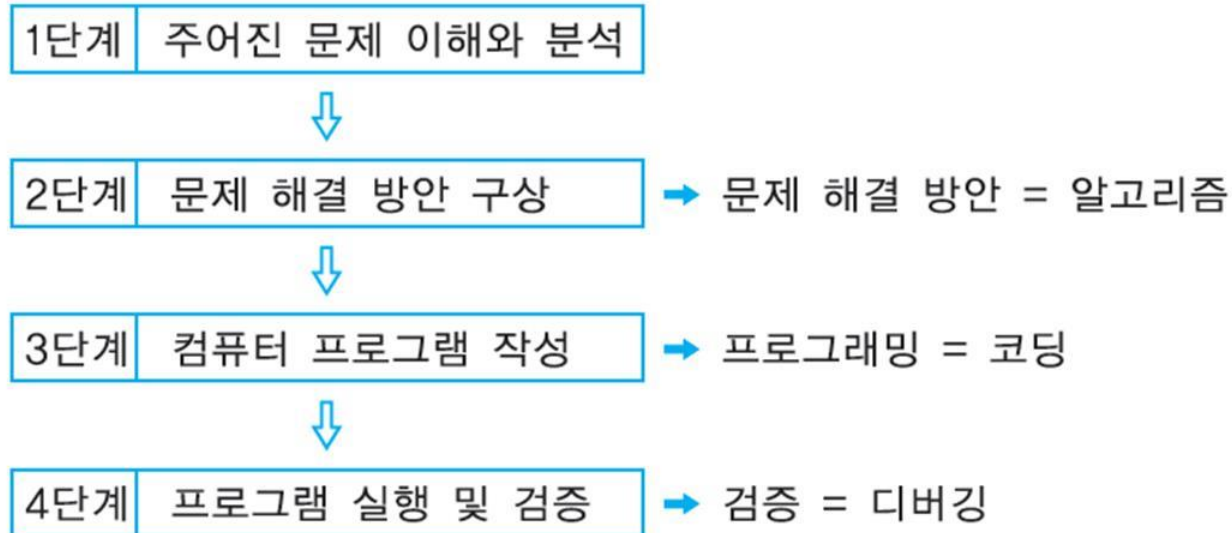
4. 응용 알고리즘 - 배열

5. 응용 알고리즘 - 자료 구조

6. 실무 응용 알고리즘

Section 1 알고리즘과 순서도

(1) 컴퓨터를 이용한 문제해결 4단계



(2) 알고리즘과 순서도

- 알고리즘을 표현하는 여러 방식 중에 순서도(Flowchart)를 가장 많이 사용한다.
 - 순서도 : 미리 약속한 기호 그림을 사용하여 논리적 절차, 흐름, 처리방법 등을 표현한 것을 말한다.
 - 순서도의 장점 : 그림을 사용하므로 알고리즘의 구조나 특성을 한눈에 파악하기 쉽고 서로 오해하지 않으면서 객관적인 의사소통이 가능하다.
- 가장 좋은 알고리즘의 조건
 - 주어진 문제를 해결하기 위한 알고리즘이 한 개만 존재하는 것이 아니다.
 - 좋은 알고리즘은 우선 그 출력 결과가 100% 정확해야 할 뿐만 아니라, 이를 프로그램으로 옮겨 컴퓨터에서 실행했을 때 그 성능이 좋아야 한다.
 - 알고리즘의 성능은 시간과 공간을 기준으로 측정한다. 알고리즘을 실행할 때 걸리는 시간이 짧고, 데이터 처리에 필요로 하는 저장 공간이 작을수록 성능이 좋은 알고리즘이다.



Section 2 순서도의 기본 개념

(1) 순서도의 기본 개념

기호	이름
	단말 기호
	흐름선
	준비 기호
	처리 기호
	입출력 기호
	프린터 출력 기호
	키보드 입력 기호
	조건 기호

기호	이름
	결합 기호
	반복 기호
	설명 기호

(2) 순서도의 추가 기호

기호	이름
	서브루틴 호출 연결
	연결 기호

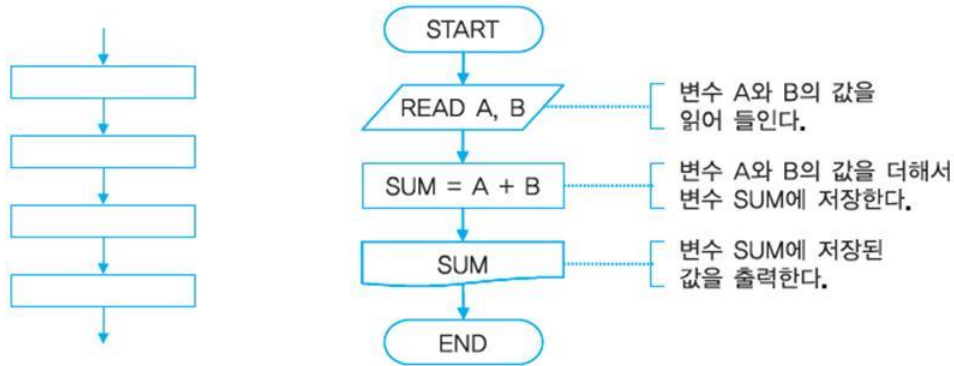
Section 3 순서도 작성법

(1) 순서도의 작성의 기본 사항

- 순서도는 시작 기호에서 출발하여 완료 기호로 마친다.
 - 시작 기호 안에는 알고리즘 이름을 기록하기도 한다.
- 기호와 기호 사이는 흐름선으로 연결하여 작업의 흐름을 명시한다.
- 흐름선의 방향은 가급적 위에서 아래로, 왼쪽에서 오른쪽으로 향하도록 한다.
 - 반복 구조 등 특별한 경우는 예외로 한다.
- 흐름선은 가급적 교차되지 않도록 하여 혼동을 피한다.
 - 둘 이상의 흐름선이 합류되어야 하는 경우에는 결합 기호를 사용한다.
- 값을 보관하고 처리하기 위하여 적절한 변수를 사용한다.
- 같은 종류의 여러 값들을 한꺼번에 보관하고 처리하기 위해서는 배열 변수를 사용한다.
- 사용할 변수(특히 배열 변수)는 준비 기호 안에 선언하며, 필요하다면 초기값도 배정한다.
- 작업 과정이 길거나 복잡하면 나누어 작성한 후 연결 기호를 사용하여 연결한다.

(2) 순서도를 구성하는 3대 기본 구조

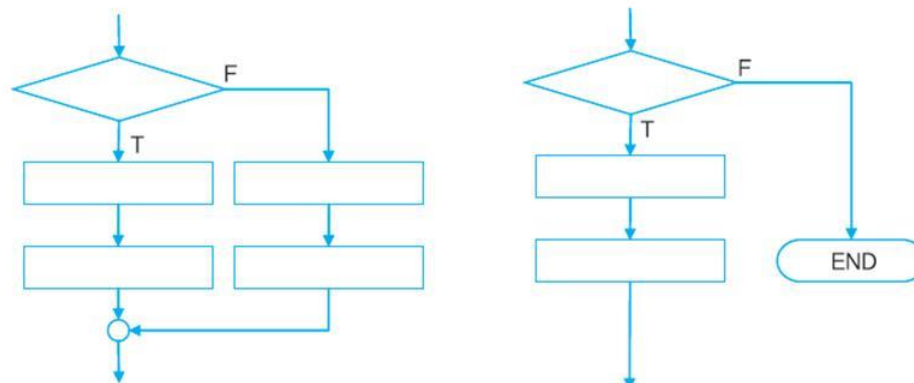
① 직선 구조 : 흐름선을 따라 위에서 아래로 차례대로 작업을 진행하는 구조



직선 구조

예 두 값을 읽은 후 합을 출력하는 알고리즘

② 분기 구조 : 조건 기호에 대한 검사 결과에 따라 진행할 다음 작업이 달라지는 구조

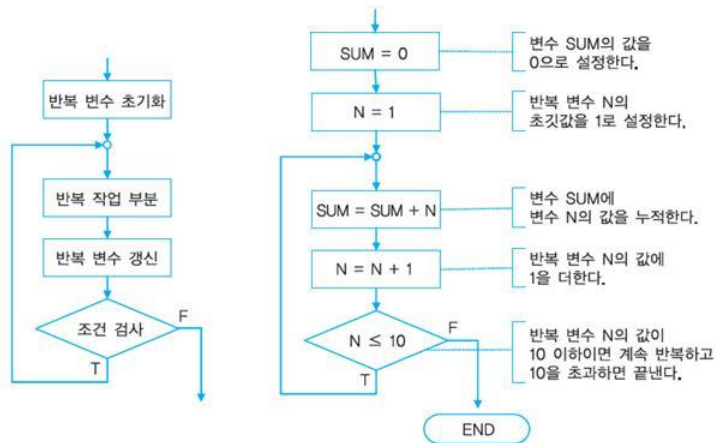


조건에 따라 수행하는
후속 작업이 다른 분기 구조

조건을 만족하지 못할 때
작업을 종료하는 분기 구조

③ 반복 구조 : 일정한 작업들을 반복하는 구조

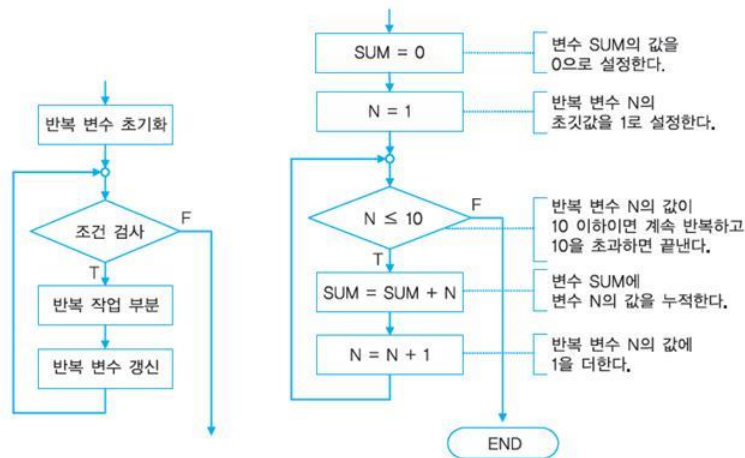
- 선처리 후검사



선처리 후검사의 반복 구조

선처리 후검사의 반복 구조를 활용하여 자연수 1부터 10까지의 합을 구하는 알고리즘

- 선검사 후처리



선검사 후처리의 반복 구조

선검사 후처리의 반복 구조를 활용하여 자연수 1부터 10까지의 합을 구하는 알고리즘

Section 4 변수와 배열

(1) 변수의 개념

- 변수는 순서도에서 값을 보관하여 처리하기 위한 장소이다.
- 모든 변수는 컴퓨터상에서 메모리 공간이 할당된다.
- 변수는 다양한 종류의 값을 보관할 수 있다. 예) 자연수, 실수, 문자, 부울린 등
- 어떤 변수를 사용하려면 순서도의 준비 기호 안에 미리 선언하여 알리는 것이 좋지만, 선언하지 않아도 변수라는 사실을 명확하게 알 수 있다면 선언을 생략할 수도 있다.
- 변수에는 초깃값이 주어질 수 있다.

(2) 변수와 배정문

순서도의 처리 기호를 통하여 변수에 값을 배정하거나 변경할 수 있다.

$A = 100$	변수 A에 값 100을 저장한다.
$B = A + 50$	변수 A의 값과 50을 더하여, 변수 B에 저장한다.
$B = B + 10$	변수 B의 현재 값에 10을 더하여, 변수 B의 새로운 값으로 저장한다(누적).

(3) 배열 변수와 준비 기호

- 배열 변수는 같은 종류의 값들을 메모리 공간 안에 연속적으로 보관하여 처리하기 위한 장소를 말한다.
- 학생 100명의 영어 점수를 보관하려면 별도의 변수 100개가 필요하지만, 배열 변수를 사용하면 1개가 필요하다. 다만 이 배열 변수는 같은 종류의 값을 보관하는 원소 100개를 갖는다.
- 학생 100명의 영어 점수와 수학 점수를 보관하려면 별도의 변수 200개가 필요하지만, 배열 변수를 사용하면 1개로도 충분하다. 다만 이 배열 변수는 같은 종류의 값을 보관하는 원소로 100개씩 총 2세트가 필요하다(한 세트는 영어 점수용, 다른 한 세트는 수학 점수용).
- 배열 변수는 순서도의 준비 기호 안에 배열 변수의 차원과 원소 개수 정보를 미리 선언한 후 사용해야 한다.

(4) 배열 변수와 반복 구조

- 배열 변수에서는 원소에 접근할 때 첨자를 이용하므로, 반복 구조에서 반복 변수를 배열 변수의 첨자로 사용하면 배열 변수의 원소에 접근하기 편리하다.
- 다음 예를 보면, 변수 K가 1차원 배열 변수 ENG의 원소에 접근하기 위한 첨자로 사용되고 동시에 반복 구조 안에서 반복 변수로도 사용된다.



예 배열 변수 ENG의 50개 원소의 값을 읽어 들이는 알고리즘

(5) 순서도 안에 등장하는 변수의 주요 기능

- **입력 변수** : 입력 기호 안에서 사용되는 변수
- **출력 변수** : 출력 기호 안에서 사용되는 변수
- **반복 변수** : 반복 구조에서 반복 여부를 결정하기 위하여 조건 기호 안에서 사용하는 변수, 반복 기호 안에 등장하는 변수
- **스위치 변수** : 값의 부호가 교대로 바뀌면서 등장하는 변수
- **임시 변수** : 어떤 값을 임시로 보관하기 위하여 처리 기호 안에 사용하는 변수

Section 5 연산과 함수

(1) 순서도에서 나올 수 있는 연산

① 처리 기호 안에 나올 수 있는 산술 연산

산술 연산자	의미	산술 연산자	의미
+	더하기	/	나누기
-	빼기	%	나머지 구하기
*	곱하기	^	거듭제곱하기

② 조건 기호 안에 나올 수 있는 관계 연산

관계 연산자	의미	관계 연산자	의미
<	작다/미만	<> 또는 ≠	같지 않음
>	크다/초과	<= 또는 ≤	작거나 같음/이하
=	같음	>= 또는 ≥	크거나 같음/이상

③ 조건 기호나 처리 기호에서 나올 수 있는 논리 연산(부울 연산)

논리 연산자	의미
AND	둘 다 True이면 True / 나머지 경우는 False
OR	둘 중 하나가 True이거나, 둘 모두가 True이면 True / 나머지 경우는 False
NOT	True는 False로 바꿈 / False는 True로 바꿈

(2) 순서도에서 나올 수 있는 함수

함수 이름	의미
ABS(X)	부호를 뺀 절대값을 계산해준다.
INT(X)	X 이하인 정수 중에서 최댓값을 찾아준다.
SQRT(X)	제곱근 값을 계산해준다.
MOD(X,Y)	X를 Y로 나눈 나머지를 계산해준다(산술 연산 %와 동일).
LEFT(X,Y)	문자열 X에 대하여 왼쪽에서부터 Y개만큼의 문자를 추출한다.
RIGHT(X,Y)	문자열 X에 대하여 오른쪽에서부터 Y개만큼의 문자를 추출한다.
MID(X,Y,Z)	문자열 X에 대하여 위치 Y에서부터 Z개만큼의 문자를 추출한다.
VAL(X)	문자열 X를 숫자로 변환한다.

Section 6 알고리즘 검증

(1) 디버깅 표 작성 방법

- 디버깅 표는 2차원 테이블로 작성된다.
- 디버깅 표의 열(세로)은 ① 값이 지정되는 변수나 ② 값을 생성하는 조건을 모아 놓는다. 즉, ① 변수에 값이 할당되는 부분이나 ② 조건 기호 안에서 값을 계산하는 부분이 여기에 등장한다.
- 디버깅 표의 행(가로)은 순서도의 진행 단계, 즉 시간 흐름을 나타낸다.
- 순서도의 각 기호마다 일련번호를 붙여 디버깅 표와 정확하게 연결될 수 있도록 한다.
- 문제 안에 배열 변수가 제공되면 디버깅 표 앞부분에 별도로 배열 변수의 원소 값들을 나열하는 것이 편리하다.
- 순서도 안에서 많은 순환이 발생하거나 많은 데이터에 대하여 반복 처리가 일어날 경우 디버깅 표가 너무 커질 수 있으므로, 규칙성이 발생하는 부분에서 생략기호(...)를 사용하여 디버깅 표의 크기를 줄일 수 있다.
- 디버깅 표는 ① 변수의 초기값을 할당하는 부분과 ② 반복 구조 등에 의해서 변수나 조건의 값이 갱신되는 부분으로 크게 구분하여 별도의 표로 만들 수도 있고, 하나의 표로 통합하여 만들 수도 있다. 보통 후자를 선택한다.