2주차 2차시 추상자료형과 알고리즘

【학습목표】

1. 추상화와 구체화를 구분할 수 있다.
2. 알고리즘의 개념과 조건을 설명할 수 있으며, 알고리즘의 표현 방법을 설명할 수 있다.

학습내용1 : 추상자료형

1. 개요 -추상화란

* 추상화 : 기억할 대상의 구별되는 특징만을 단순화하여 기억하는 기능

[그림 2-6] 뇌의 추상화 기능

2. 컴퓨터를 활용하여 어떻게 문재해결 하는가?

1) 크고 복잡한 문제를 단순화시켜 쉽게 해결

2) 자료 추상화(Data Abstraction)

<처리할 자료, 연산, 자료형에 대한 추상화된 기본 개념>

3) 자료 추상화(Data Abstraction)

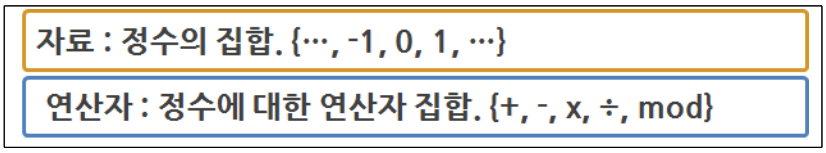
① 자료
- 프로그램의 처리 대상이 되는 모든 것을 의미
- 어떤 값(Value) 자체 의미

② 연산
- 어떤 일을 처리하는 과정. 연산자에 의해 수행

* 예시 : 더하기 연산은 +연산자에 의해 수행

③ 자료형
- 처리할 자료의 집합과 자료에 대해 수행할 연산자의 집합

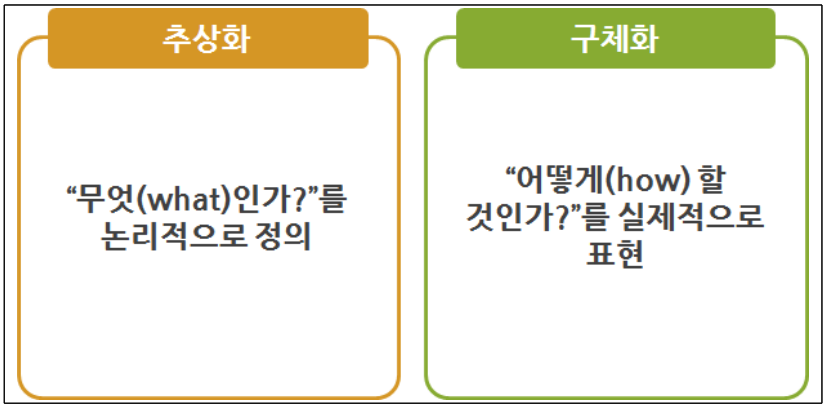
* 예시 : 정수 자료형



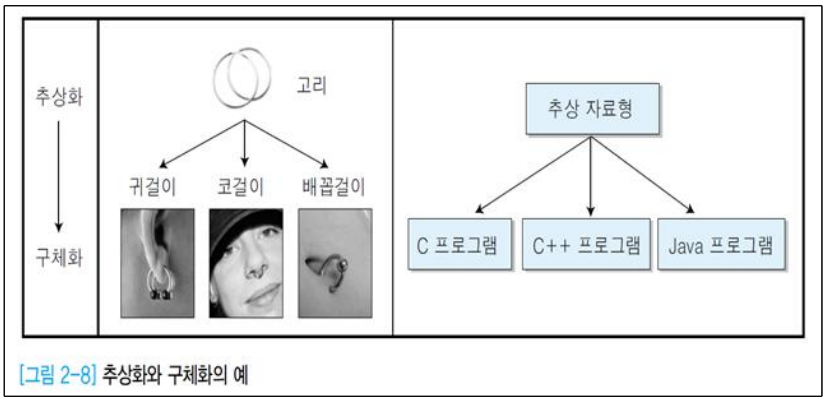
3. 추상 자료형(ADT, Abstract Data Type)

* 추상 자료형이란? : 자료와 연산자의 특성을 논리적으로 추상화하여 정의한 자료형
- 알고리즘 개발이 단순, 프로그램으로 구체화가 용이

4. 추상화와 구체화



1) 추상화와 구체화의 예



[그림 2-8] 추상화와 구체화의 예

2) 자료와 연산에 있어서의 추상화와 구체화의 관계

	자료	연산
추상화	추상자료형	알고리즘 정의
구체화	자료형	프로그램 구현

학습내용2 : 알고리즘

1. 알고리즘(Algorithm)의 이해

* 알고리즘 : 문제해결방법을 추상화하여 단계적 절차를 논리적으로 기술해 놓은 명세서

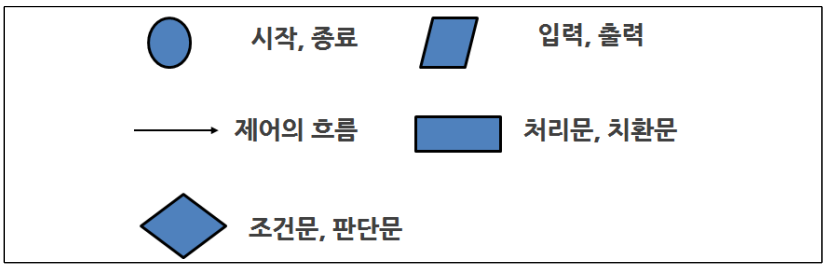
1) 알고리즘의 조건
① 입력(Input) : 알고리즘 수행에 필요한 자료가 외부에서 입력되어야 함
② 출력(Output) : 알고리즘 수행 후 하나 이상의 결과를 출력해야 함
③ 명확성(Definiteness) : 수행할 작업의 내용과 순서를 나타내는 알고리즘의 명령어들은 명확하게 명세되어야 함
④ 유한성(Finiteness) : 알고리즘은 수행 뒤에 반드시 종료되어야함
⑤ 효과성(Effectiveness) : 알고리즘의 모든 명령어들은 기본적이며 실행이 가능해야 함

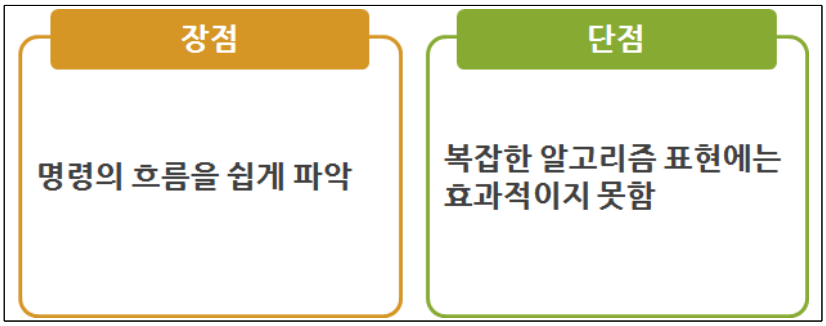
2. 알고리즘의 표현 방법

① 자연어를 이용한 서술적 표현 방법
② 순서도(Flow chart)를 이용한 도식화 표현 방법
③ 프로그래밍 언어를 이용한 구체화 방법
④ 가상코드(Pseudo-code)를 이용한 추상화 방법

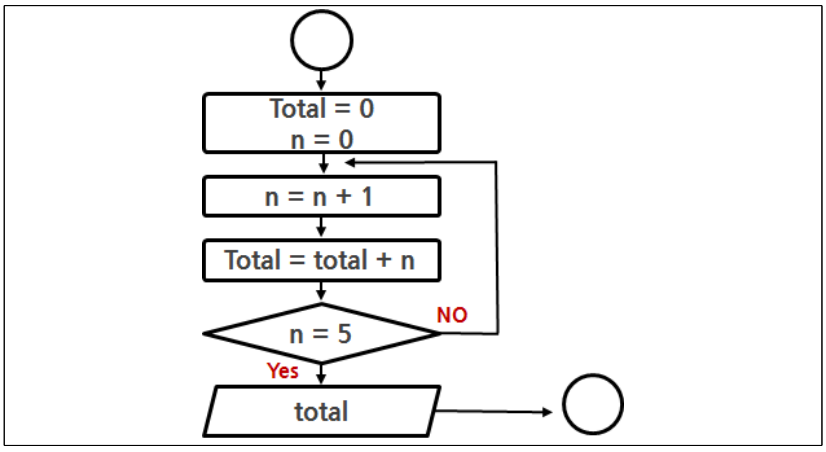
3. 순서도를 이용한 알고리즘의 표현

1) 순서도에서 사용하는 기호

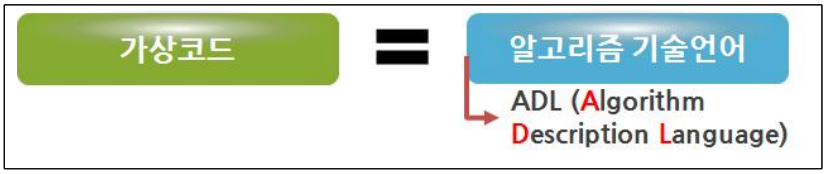




2) 순서도의 예) 1부터 5까지의 합을 구하는 알고리즘



4. 가상코드를 이용한 알고리즘의 표현



- 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 형식

1) 기본 요소

① 기호
- 문자나 숫자의 조합. 첫 문자는 반드시 영문자 사용
- 변수, 자료형 이름, 프로그램 이름, 레코드 필드 명, 문장의 레이블 등을 나타냄

② 자료형
- 정수형과 실수형의 수치 자료형, 문자형, 논리형, 포인터, 문자열 등의 모든 자료형 사용

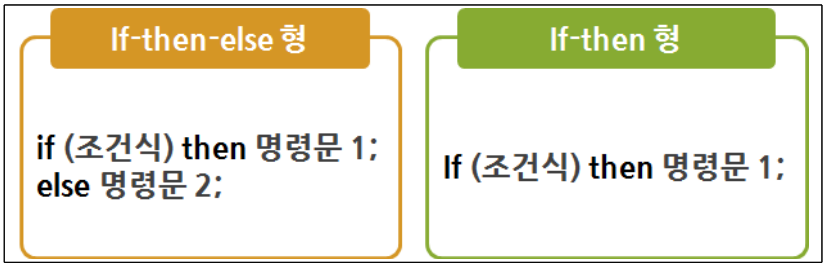
③ 연산자
- 산술연산자, 관계연산자, 논리연산자

2) 지정문
- 사용형식 : 변수 ← 값
- 지정연산자(←)의 오른쪽에 있는 값(또는 식의 계산 결과 값이나 변수의 값)을 지정연산자(←)의 왼쪽에 있는 변수에 저장

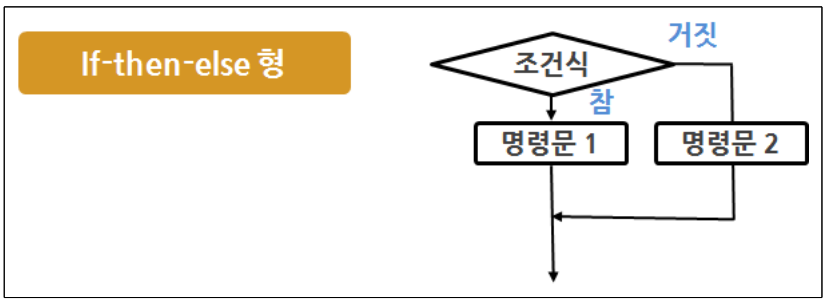
3) 조건문

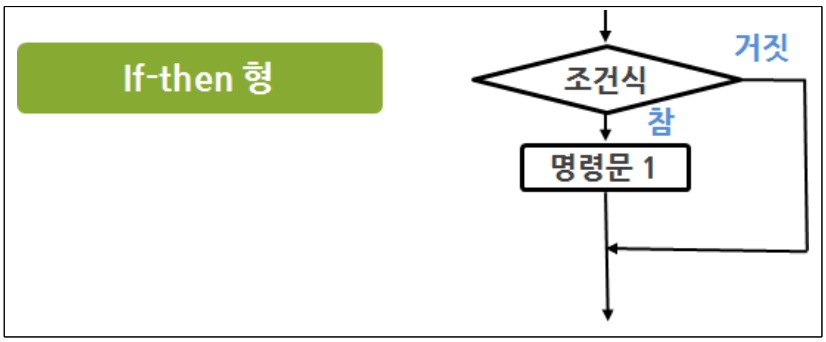
* 조건문의 정의 : 조건에 따라 실행할 명령문이 결정되는 선택적 제어구조를 만듦

① If 문 - 형식



② If 문 - 제어 흐름

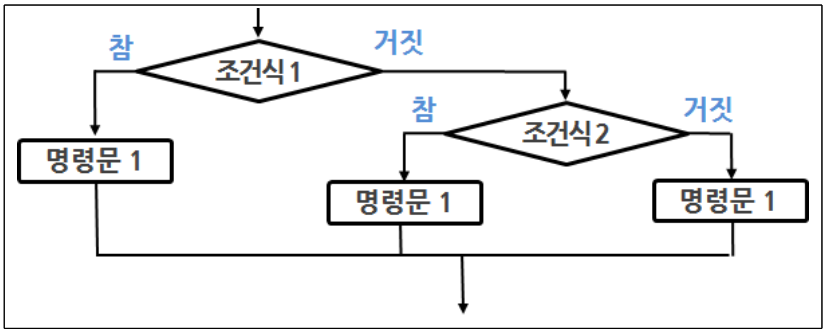




③ 중첩 If 문

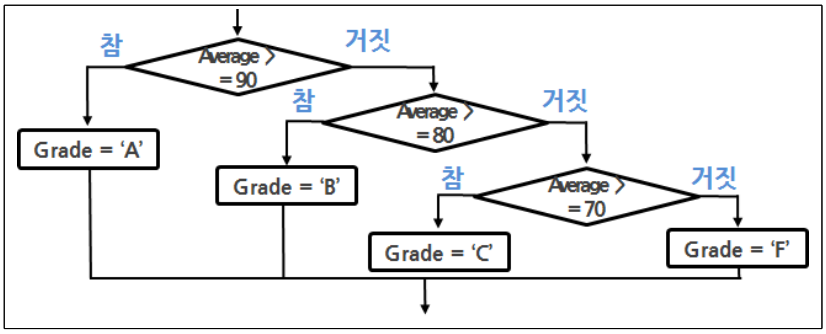
```
If (조건식 1) then 명령문 1;
else if (조건식 2) then 명령문 2;
else 명령문 3;
```

④ 중첩 if 문의 제어 흐름



⑤ 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" ;
```



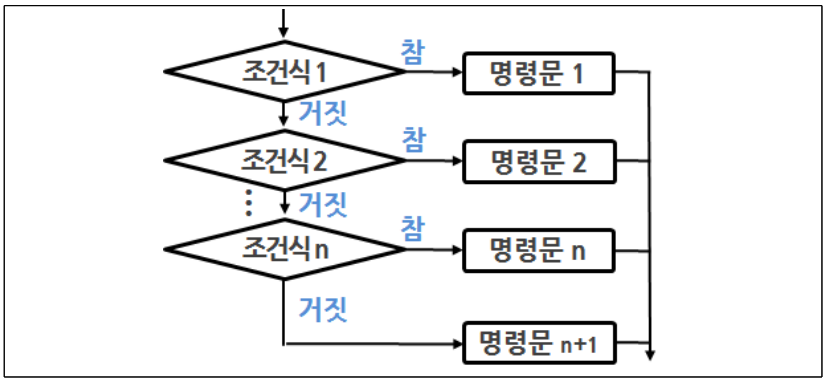
⑥ case 문

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

형식

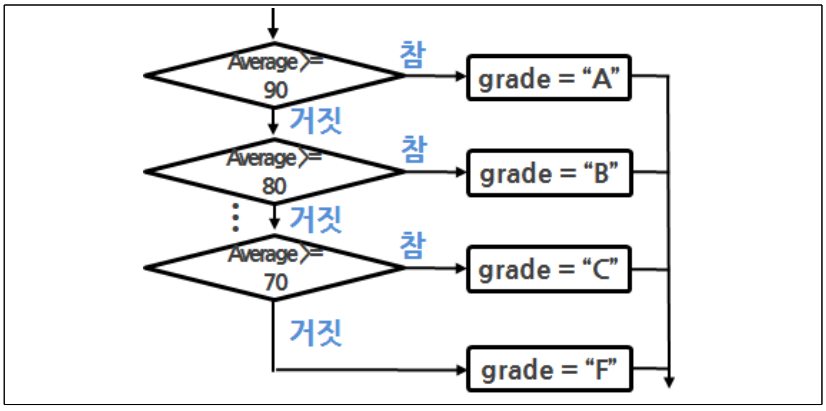
```
Case {
조건식 1 : 명령문 1;
조건식 2 : 명령문 2;
…
조건식 n : 명령문 n;
else : 명령문 n+1;
```

⑦ case 문의 제어 흐름



⑧ case 문 사용 예) 평균 점수에 따른 등급 계산하기

```
Case {
        Average >= 90 : grade = “A”;
        Average >= 80 : grade = “B”;
        Average >= 70 : grade = “C”;
        else : grade = “F” ;
}
```



4) 반복문

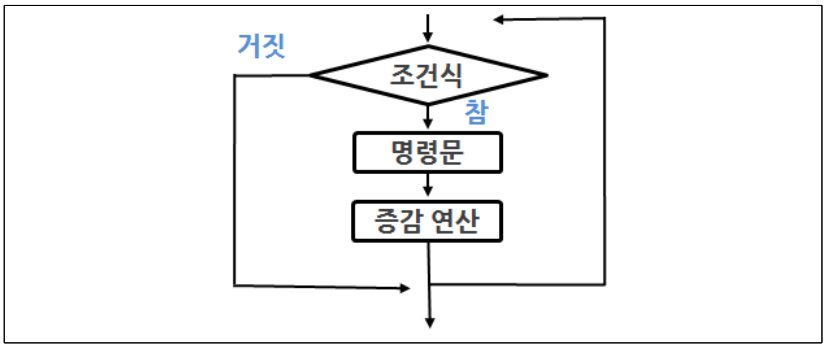
- 일정한 명령을 반복 수행하는 루프(Loop) 형태의 제어구조

① for 문

```
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

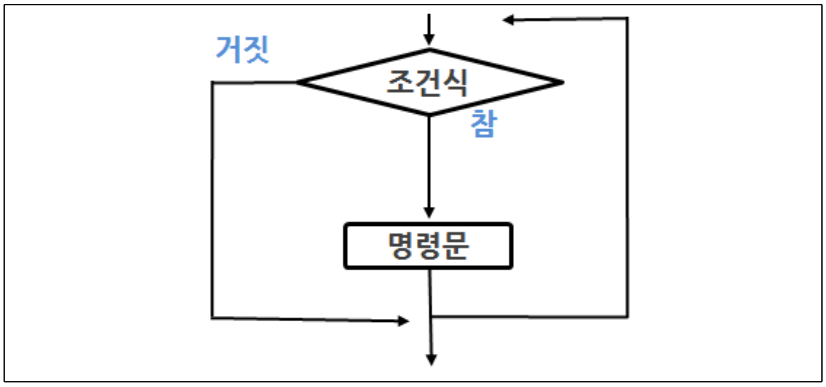
② for 문의 제어 흐름



③ While 문

```
형식   while (조건식) do 명령문 ;
```

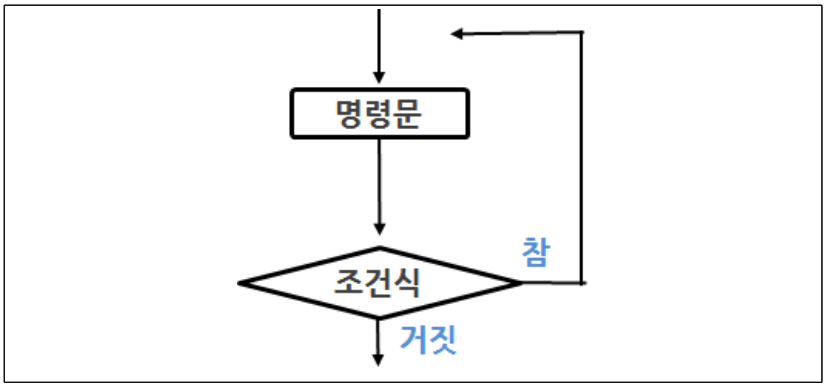
- 조건식이 참인 동안 명령문을 반복 수행
- While 문의 제어 흐름



④ do-while 문

```
형식
do 명령문;
   while (조건식);
```

- 일단 명령문을 한번 수행한 다음 조건식을 검사하여 조건식이 참이면 반복 수행
- do-while 문의 제어 흐름

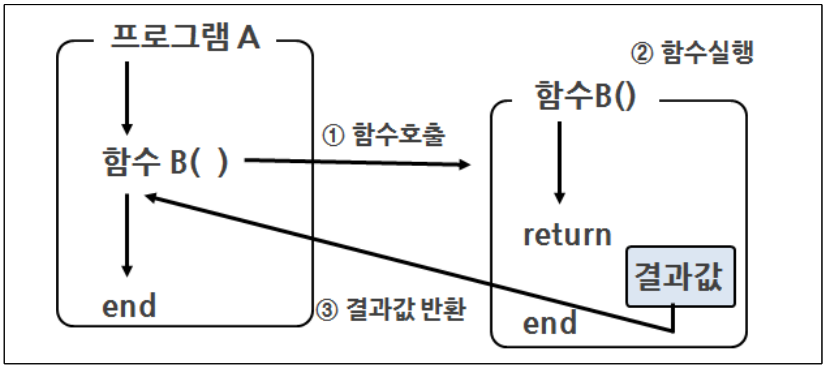


5) 함수문

- 처리작업 별로 모듈화하여 만든 단위 프로그램
- 프로그램 크기가 줄고 수정, 관리가 용이하며 재사용도 가능

```
형식
함수이름 (매개변수)
   명령문;
   ...
   return 결과값;
end
```

* 함수의 호출과 실행 및 결과값 반환에 대한 제어 흐름



【학습정리】

1. 추상화는 무엇인가를 논리적으로 정의하는 것이다.

2. 구체화는 어떻게 할 것인지를 실제적으로 표현하는 것이다.

3. 알고리즘은 주어진 문제를 해결하기 위한 방법을 추상화하여 일련의 단계적 절차를 논리적으로 기술해 놓은 명세서이다.