소프트웨어와 문제해결

Dr. Young-Woo Kwon

수업 개요

- 정렬
  - 버블 정렬, 선택 정렬
- 프로그래밍
  - 논리 & 명제 찾기
  - 알고리즘
  - 순서도 & 의사코드
- 실습

Activity 1

- 정수형 값을 2진수로 변환하고, ,변환된 2진수의 0과 1의 개수를 출력하는 프로그램을 작성하시오.
  - 입력: 7
  - 출력: 111, 1의 개수: 3, 0의 개수: 0

(복습) 선형 탐색

- 숫자들의 리스트가 있다고 가정
- 이 숫자들은 정렬되어 있을 수도 있으며 아닐 수도 있음 아직 정렬되었는지 모름
- 선형 탐색은 리스트에서 순차적으로 비교하면서 숫자를 찾는 방법

→ 순차적으로 검색

3	5	2	1	0	9	7	8	6	4

(복습) 이진 탐색

- 숫자들의 리스트가 있다고 가정
- 이 숫자들은 크기 순으로 정렬되어 있음
- 이진 탐색은 리스트를 반으로 나누어 가운데 숫자와 비교하며 원하는 숫자를 찾는 알고리즘

(복습) 해시 탐색

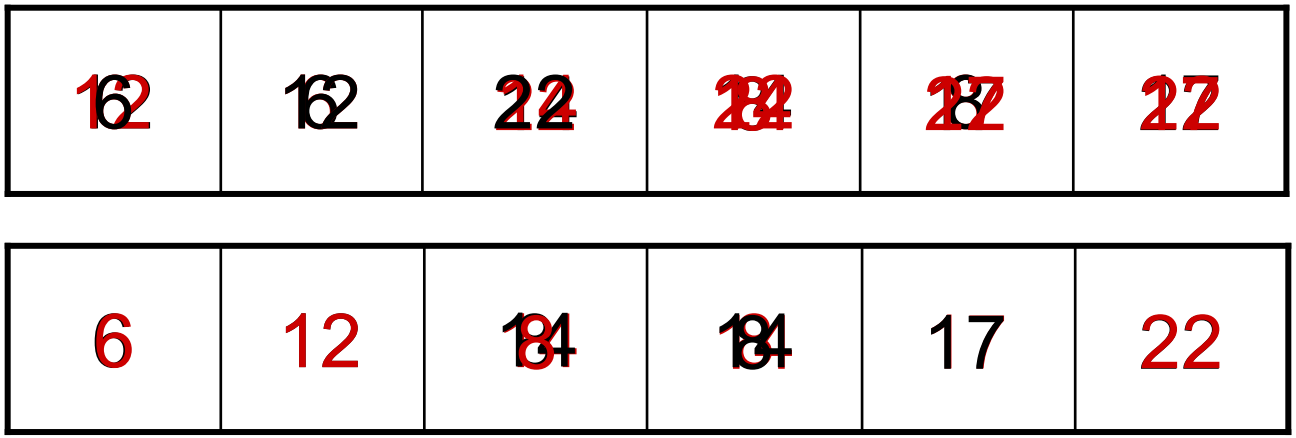
- 숫자들의 리스트가 있다고 가정
- 이 숫자들은 그룹으로 나누어져 저장되어 있음
- 해시 탐색은 해당하는 숫자가 속한 그룹을 순차적으로 비교하여 원하는 숫자를 찾는 알고리즘이다

Activity 2

- 두 변수에 정수형 값을 할당하고 이를 교환하는 코드를 작성하시오 (변수의 이해)
  - a = 2, b = 3이라면 결과는 a = 3, b = 2 임

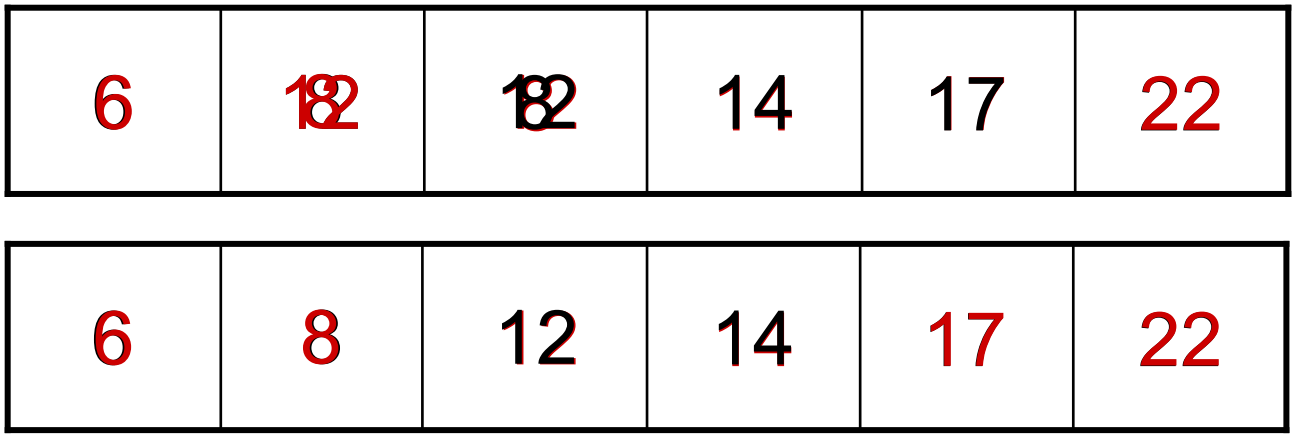
정렬

Bubble Sort



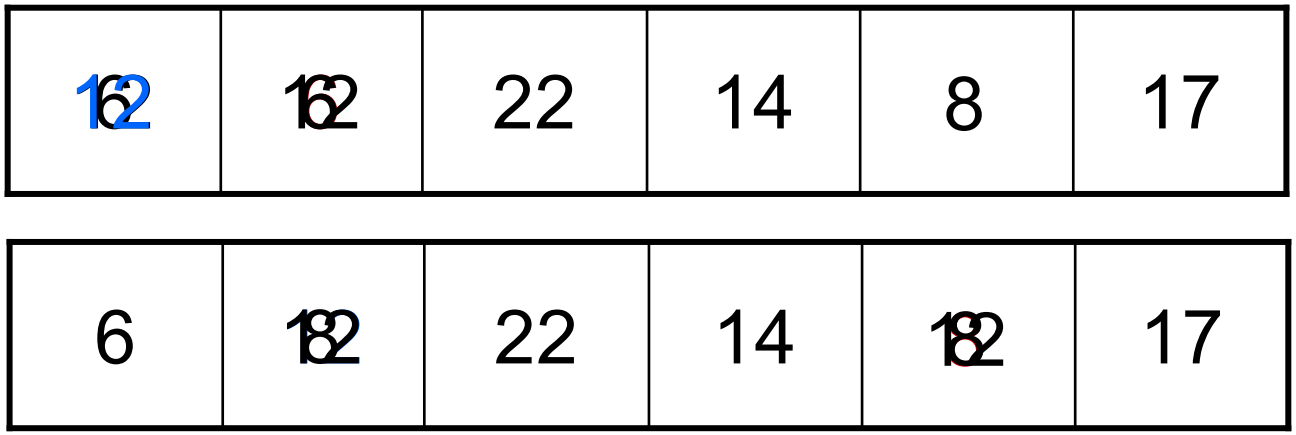
- Given n numbers to sort:
- Repeat the following n-1 times:
  - For each pair of adjacent numbers:
    - If the number on the left is greater than the number on the right, swap them.

Bubble Sort



- Given n numbers to sort:
- Repeat the following n-1 times:
  - For each pair of adjacent numbers:
    - If the number on the left is greater than the number on the right, swap them.

Selection Sort



- Given n numbers to sort:
- Repeat the following n-1 times:
  - Mark the first unsorted number
  - Find the smallest unsorted number
  - Swap the marked and smallest numbers

Selection Sort

- Given n numbers to sort:
- Repeat the following n-1 times:
  - Mark the first unsorted number
  - Find the smallest unsorted number
  - Swap the marked and smallest numbers

논리란?

- 논리 (logic)
  - 인간의 추론을 논리 시스템으로 정의
  - 올바른 사고의 과학 (Science of correct thinking)
  - 두개의 범주: 귀납적 논리, 연역적 논리
- 귀납적 논리 (inductive logic)
  - 관찰 또는 경험으로부터 어느 정도의 확실성을 가지고 결론 도출
  - 도출된 결론은 필연적으로 불확실
    - 결론이 기반을 두고 있는 경험들의 횟수와 직접적으로 관련하기 때문
    - 예) 방울양배추를 2번 먹은 후 병이 났다
      - 결론: 방울양배출에 알레르기가 있다.
      - 방울양배추를 먹을 때마다 병이 났었다면, 결론의 확실성이 증가

      → 경험의 횟수에 따라 결론의 확실성 증가

논리란?

- 연역적 논리 (deductive logic)
  - 확실하다고 생각되는 것은 절대적으로 참이라는 가정으로 시작
  - 확실하다고 생각되는 것으로부터 다른 사실들도 절대적으로 참이라고 생각
  - 가정이 참이라고 한다면 결론은 확실한 참
    - 예) 모든 인간은 죽는다. 아리스토텔레스는 인간이다. 그러므로 아리스토텔레스는 죽는다

논리란?

- 기호 논리(symbolic logic)
  - 진실에 대한 진술을 표현하기 위해 기호를 사용
- 논리 연산자
  - 논리적 사고를 표현
- 부울 논리(boolean logic)
  - 수학자 조지 부울에 의해 만들어진 기호 논리 시스템
  - 현대의 컴퓨팅 시스템은 올바르고 신뢰할 결과를 생성하기 위해 부울 논리 규칙에 의존함

부울 논리

- 명제(proposition)
  - 부울 논리의 기본 단위
  - 참 또는 거짓이 될 수 있는 진술
    - 예) “에베레스트 산이 세상에서 가장 높은 산이다.”
    - 기본 또는 합성 둘 중의 하나가 될 수 있음
- 논리 값(logical values)/진리값(truth values)
  - 부울논리의 사용 값: 참 또는 거짓
- 배중률(law of the excluded middle)
  - 논리시스템에서 참 또는 거짓인 진리 값만 존재

부울 논리

- 기본 명제(simple proposition)
  - 부분들로 쪼개질 수 없는 명제
- 합성 명제(compound proposition)
  - 기본 명제들을 논리 연산자로 결합
  - 논리 연산자 :(and, or, implies, not)
- 명제를 약어로 사용
  - **P** = "I am hungry.", **Q** = "I am cold."
- 논리 연산자를 사용
  - "I am hungry and I am cold." → P **and** Q

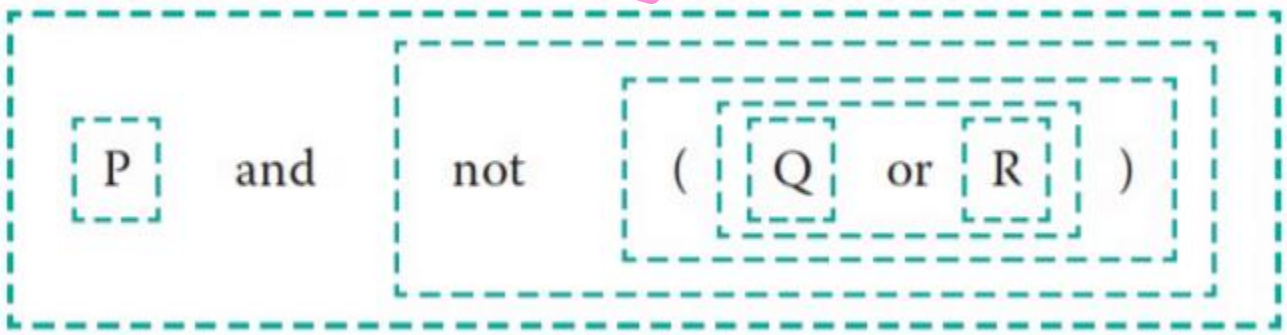
P and Q
→ 합성 명제

제대로 된 명제 작성하기

- 잘 작성(well formed)된 명제
  - 명제가 의미를 가지기 위해 잘 작성되어야 함
  - 다음의 규칙을 따름
- 규칙 1 – 다음의 각각은 기본 명제이다.
  - a. 어떤 단일 문자
  - b. 참
  - c. 거짓
- 규칙 2 – 박스(□)가 박스가 어떤 명제라고 가정, 다음의 각각 또한 명제이다.
  - a. □ and □
  - b. □ or □
  - c. □ implies □
  - d. □ ≡ □
  - e. Not □
  - f. (□)

제대로 된 명제 작성하기

- P and not (Q or R)은 잘 작성된 명제인가?
  - P and not (Q or R) → □ and not (□ or □)(규칙 1a)
  - □ and not (□ or □) → □ and not (□)(규칙 2b)
  - □ and not (□) → □ and not □(규칙 2f)
  - □ and not □ → □ and □(규칙 2e)
  - □ and □ → □(규칙 2a)
  - **따라서 주어진 명제는 잘 작성되었고 의미 있는 명제**



제대로 된 명제 작성하기

- 문자들과 연산자들의 일부 조합은 의미있는 부울 명제처럼 보이지만 실제로 기호들의 의미 없는 혼합
- P not Q은 잘 작성된 명제 인가?
  - P not Q → □ not □ (규칙 1)
  - □ not □ → □□ (규칙 2e)
  - □□을 단일 박스로 축소해줄 어떤 규칙도 없음

  → 결론: "P not Q"는 명제가 아니다

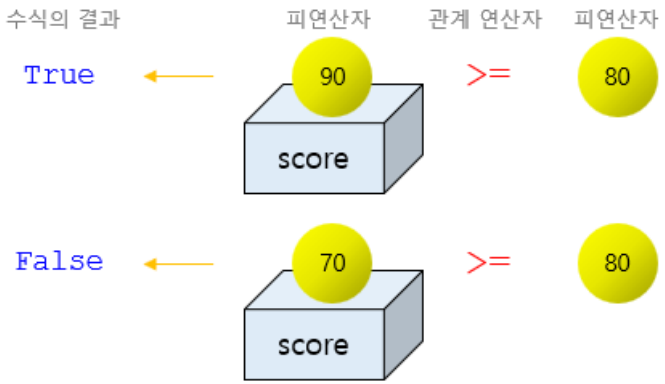
명제 판별하기

- 명제판별의 목적: 잘 작성되었는지가 아니고 명제의 진리값을 결정하는 것
- 연산자(operator): 입력 받고, 입력값을 처리하고, 단일 출력값을 만들어내는 기계같은 것
- 연산자의 애리티(arity): 연산자에 대한 입력의 개수
- 이항 연산자(binary operator): 애리티가 2인 연산자
- 단항 연산자(unary operator): 애리티가 1인 연산자

연산자	이름	애리티	사용 예
and	논리곱	이항	P and Q
or	논리합	이항	P or Q
implies	함축	이항	P implies Q
≡	동치	이항	P ≡ Q
not	논리적 부정	단항	not P

관계 연산자

- 관계 연산자(relational operator)
  - 두 개의 피연산자를 비교하는 데 사용
  - 관계 연산자 수식의 결과는 참(True)/거짓(False)으로 계산
  - 관계 연산자가 사용된 조건 수식
    - '점수가 80 이상인' 문장은 'socre >= 80' 조건 수식으로 나타냄
    - 만약 변수 score의 값이 90일 경우 수식의 결과는 참(True)이 되고, 70일 경우 수식의 결과는 거짓(False)이 됨



관계 연산자

- 관계 연산자

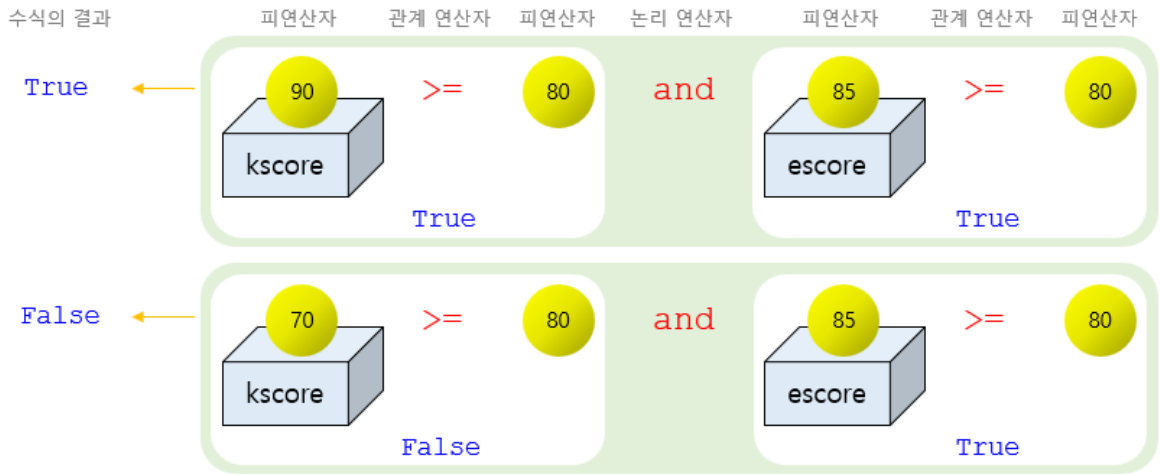
관계 연산자	의미	결과 (x:6, y:2 인 경우)
x > y	x가 y보다 큰가?	True
x >= y	x가 y보다 크거나 같은가?	True
x < y	x가 y보다 작은가?	False
x <= y	x가 y보다 작거나 같은가?	False
x == y	x와 y가 같은가?	False
x != y	x와 y가 다른가?	True

= 연산자와 == 연산자의 구분

프로그램의 코딩 작업에서 = 연산자와 == 연산자를 혼동하거나 잘못 사용하여 작성하는 경우가 많다.
= 연산자는 대입 연산자이고, == 연산자는 관계 연산자이다.

논리 연산자

- 논리 연산자(logical operator)
  - and 연산자
    - '국어 점수가 80점 이상이고 영어 점수가 80점 이상이면'
      'kscore >= 80' 조건과 'escore >= 80'
      'kscore >= 80 and escore >= 80'

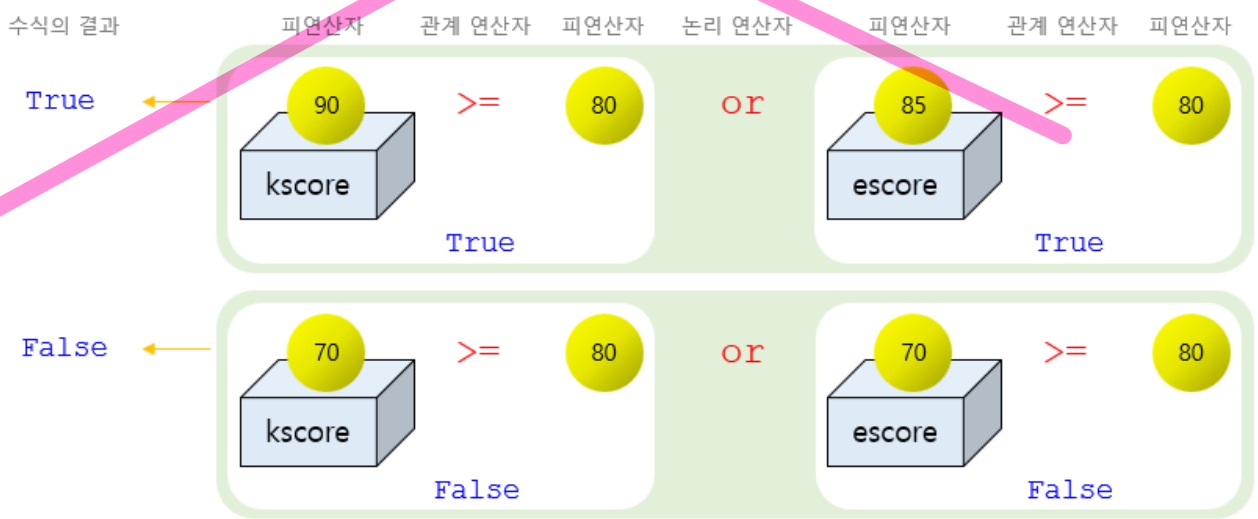


논리 연산자

- 논리 연산자(logical operator)
  - or 연산자
    - '국어 점수가 80점 이상이거나 영어 점수가 80점 이상이면

      'kscore >= 80' 조건 혹은 'escore >= 80'

      'kscore >= 80 or escore >= 80'



논리 연산자

- 논리 연산자(logical operator)

논리 연산자	의미
x and y	x와 y가 모두 True이면 True, 그렇지 않으면 False
x or y	x나 y중에서 하나만 True이면 True, 모두 False이면 False
not x	x가 True이면 False, x가 False이면 True

  - 진리표(True table)
    - 진리식/논리식/논리회로에 대한 입출력 결과를 기록한 표

x	y	x and y	x or y	not x
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

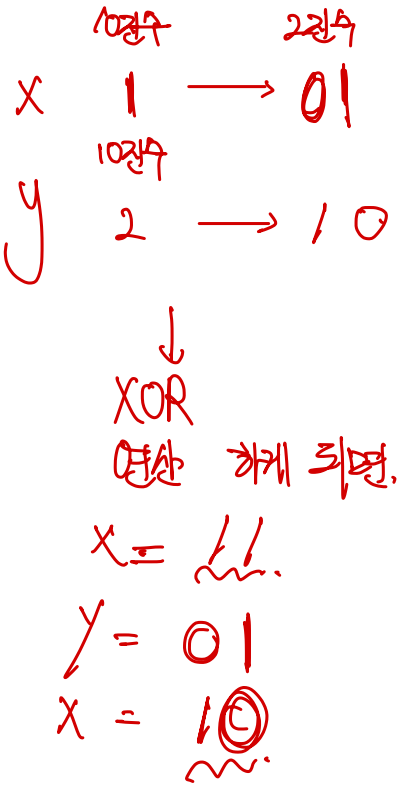
논리 연산자

- XOR
  - 입력들의 값이 같지 않으면 참
  - 0 XOR 0 = 0
  - 0 XOR 1 = 1
  - 1 XOR 0 = 1
  - 1 XOR 1 = 0

Swap

- 다음 코드의 실행 결과는?

```
x = 1
y = 2
print(x, y)

x = x ^ y
y = y ^ x
x = x ^ y
print(x, y)
```



Swap

```
x = 1
y = 2
print(x, y)

x = x ^ y
y = y ^ x
x = x ^ y
print(x, y)
```

01 XOR 10 → { X = 11, Y = 10 }
10 XOR 11 → { X = 11, Y = 01 }
11 XOR 01 → { X = 10, Y = 01 }

Q: 3, 4

{011, 100} → {111, 100} → {111, 011} → {100, 011}

알고리즘

- 문제 해결 방법을 컴퓨터가 이해할 수 있는 언어로 기술한 것 → 명령어들의 집합
- 다음 조건들을 만족하여야 함
  - 입/출력
  - 명백성
  - 유한성
  - 유효성

소프트웨어와 프로그래밍 언어

- 소프트웨어는 프로그래밍 언어로 작성
  - 프로그래밍 언어는 알고리즘을 정확하고 간결하게 표현할 수 있도록 설계된 언어
- 컴퓨터는 프로그래밍 언어로 작성된 단계들을 그대로 수행만 함
- 프로그래밍 언어의 종류
  - 명령형 언어
    - 명령어를 순차적으로 실행
  - 함수형 언어
    - 입력값과 출력값의 관계로 프로그램 표현
  - 선언형 언어
  - 입력값에 연관있는 사실을 사용하여 출력값을 생성하도록 표현함

프로그래밍

- 모든 실행 가능한 알고리즘은 □ □ 가능한 동작들의 순서로 표현
  - 컴퓨터 (computable) 동작 : 컴퓨터가 실행할 수 있는 동작
  - 계산 가능한 함수 (computable functions) : 컴퓨터가 실행할 수 있는 함수
- 값 이름 연동 (binding)
  - 이름(식별자, identifier, variable)에 값을 연결시키는 것

```
X ← 3
Y ← 4
Z ← 3+ 4
```

```
X ← 3
Y ← 4
Z ← X + Y
```

상태 (State)

- 프로그램이 실행됨에 따라 연동 상태가 변하게 된다

```
X ← 3
X ← 4
X ← X+ X
```

식별자 X의 값이 프로그램이 실행됨에 따라 3, 4, 8으로 바뀐다

- 계산 상태 (computational state)
  - 실행 중 어느 한 순간의 활성화된 값 이름 연동의 집합

계산 상태

1. X ← 3

{ X = 3, Y = ? }

2. Y ← X + 4

{ X = 3, Y = 7 }

3. Y ← X + Y

{ X = 3, Y = 10 }

4. X ← X * Y

{ X = 30, Y = 10 }

프로그램

계산 상태

계산 상태 예

- 섭씨 온도 → 화씨 온도 프로그램

섭씨에서 화씨로 바꾸는 알고리즘

1. Celsius ← 33.5
2. Fahrenheit ← Celsius * 9
3. Fahrenheit ← Fahrenheit / 5
4. Fahrenheit ← Fahrenheit + 32

섭씨 온도에서 화씨 온도로 바꾸기

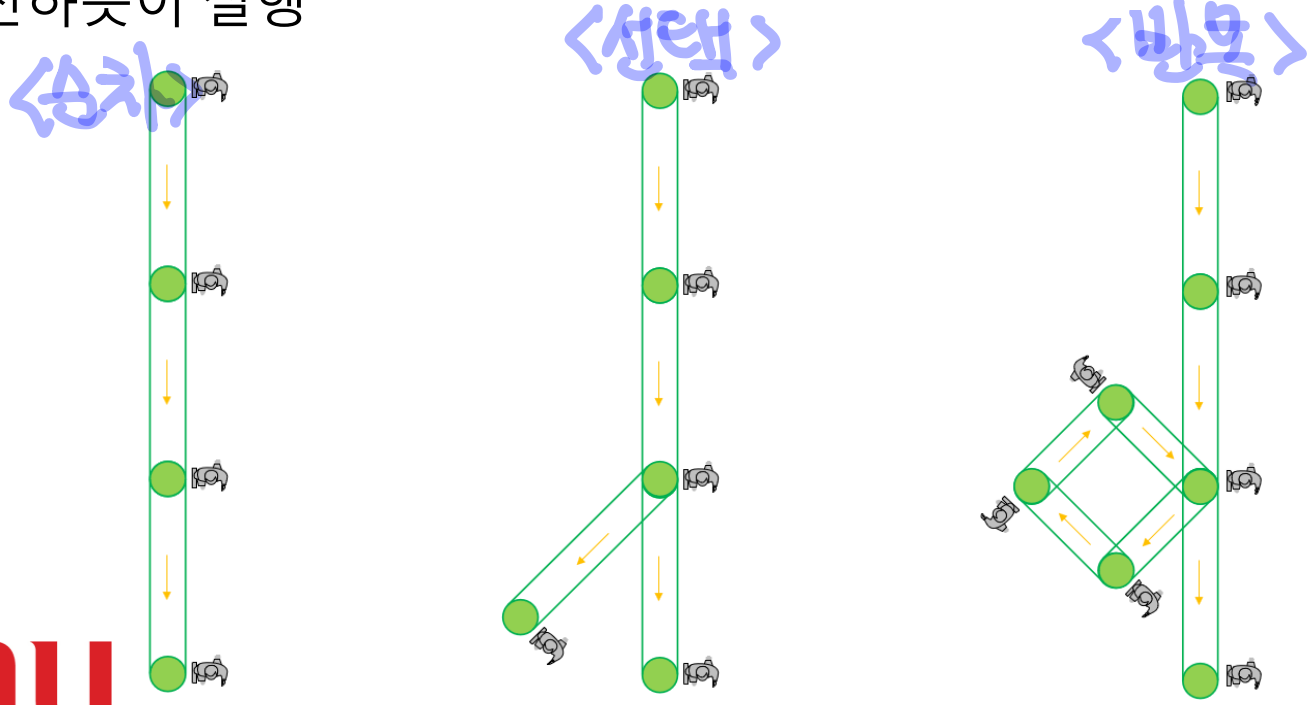
1. Celsius ← 33.5
   현재 상태는 {Celsius = 33.5}
2. Fahrenheit ← Celsius * 9
   현재 상태는 {Celsius = 33.5 and Fahrenheit = 301.5}
3. Fahrenheit ← Fahrenheit / 5
   현재 상태는 {Celsius = 33.5 and Fahrenheit = 60.3}
4. Fahrenheit ← Fahrenheit + 32
   현재 상태는 {Celsius = 33.5 and Fahrenheit = 92.3}

순서도

- 알고리즘에서의 작업 순서를 그림으로 표현
- 순서도의 단점
  - 한번 작성된 순서도는 변경하기가 어려움
  - 알고리즘이 복잡하면 기술하기가 어려움

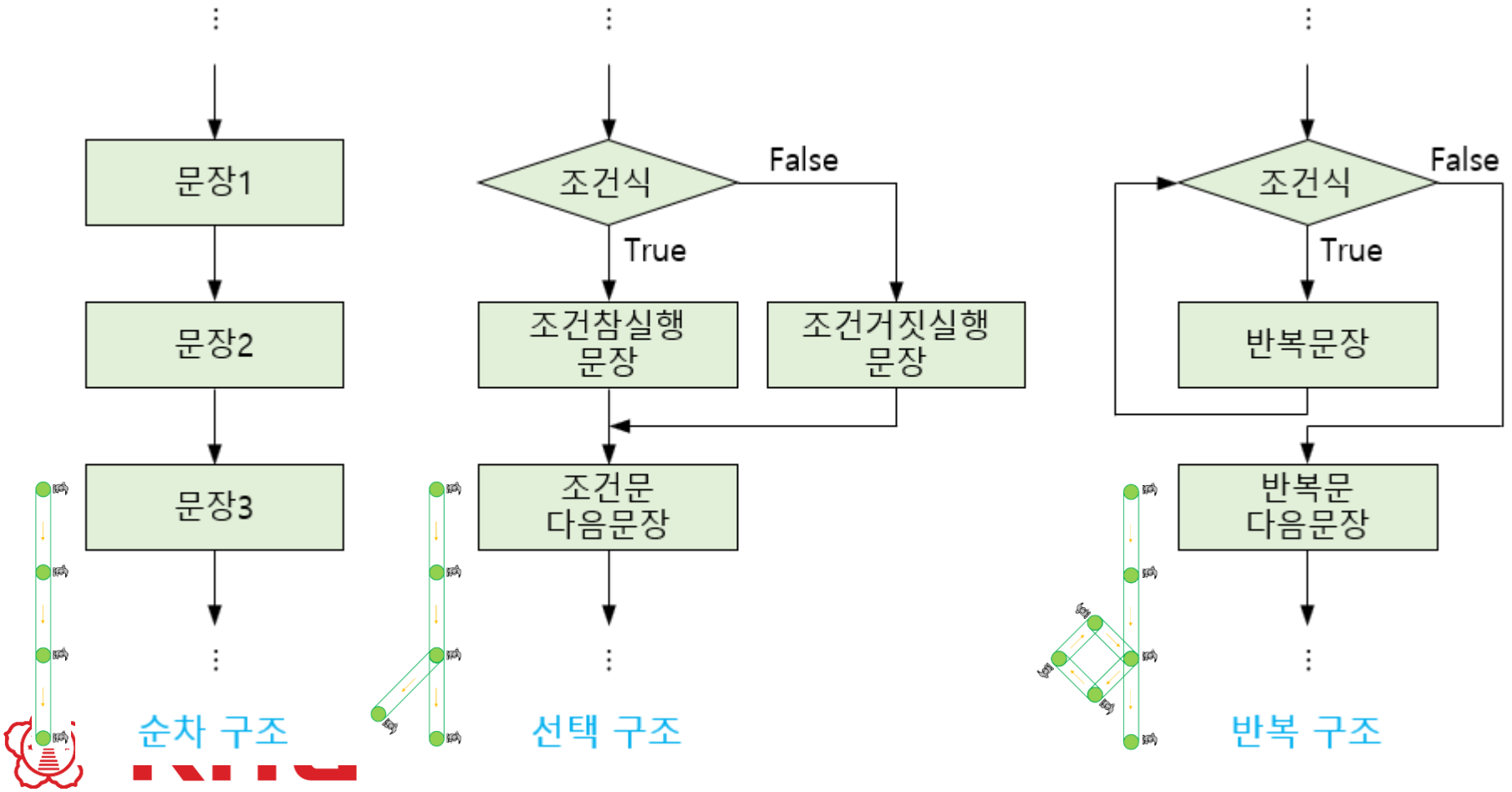
프로그램 실행 흐름 구조

- 기본 제어 구조
  - 순차(sequence) 구조 : 위에서 아래로 차례대로 실행
  - 선택(selection) 구조 : 조건에 따라 선택하여 실행
  - 반복(iteration) 구조 : 동일한 문장이나 부분이 여러 번 반복되어 회전하듯이 실행



프로그램 실행 흐름 구조

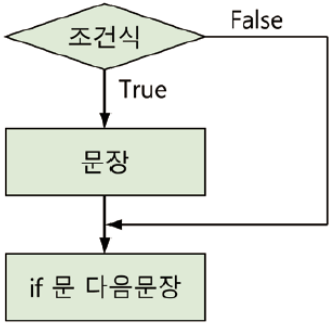
- 기본 제어 구조



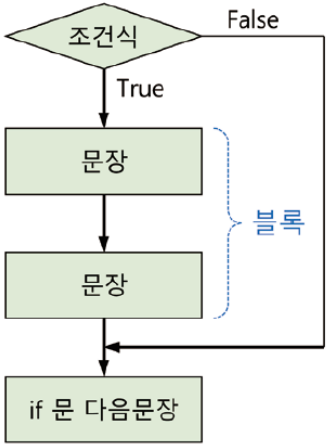
조건에 만족하면 실행하기

- if 문
  - 선택 구조를 위한 기본적인 문장
  - if 문 내에 조건식을 포함하여 조건문이라고 함
  - 조건에 맞으면(조건식의 값이 참(True)이면) 문장/블록을 실행하고 그렇지 않으면 건너뜀

```
if 조건식:
    문장
```




```
if 조건식:
    블록
```

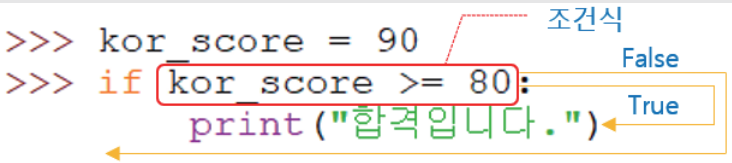


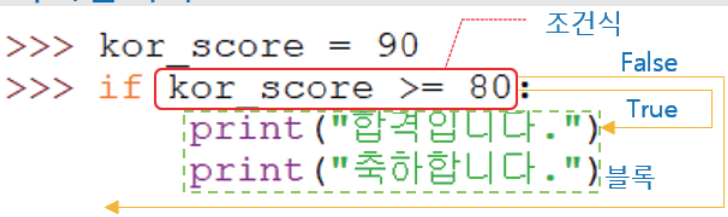
조건에 만족하면 실행하기

- if 문
  - 조건식
    - 관계 연산자나 논리 연산자 등이 사용된 수식
    - 조건식의 결과는 참(True)이나 거짓(False)으로 변환됨
    - 조건식의 값이 참이면 문장이나 블록이 실행되고,
      조건식의 값이 거짓이면 문장이나 블록을 실행하지 않고 건너뜀

```
>>> kor_score = 90
>>> if kor_score >= 80:
        print("합격입니다.")

합격입니다.
```

```
>>> kor_score = 90
>>> if kor_score >= 80:
        print("합격입니다.")
        print("축하합니다.")

합격입니다.
축하합니다.
```





조건에 만족하면 실행하기

- if 문
  - 블록(block)
    - 여러 개의 문장들을 하나의 공간 안에 모아둔 것
    - 하나의 블록 안에 속한 여러 개의 문장들은 모두 같이 실행됨
    - 블록에 있는 문장들은 그 위의 문장들과 비교할 때 기본적으로 4칸의 공백으로 들여쓰기를 하며, 이 공백의 개수에 의해 블록에 속했는지 여부를 판별하게 됨
    - 블록 내에서 들여쓰기를 한 공백의 개수가 서로 다르면 오류 발생

```
>>> kor_score = 90
>>> if kor_score >= 80:
        print("합격입니다.")
        print("축하합니다.")

SyntaxError: unexpected indent
```

```
>>> kor_score = 90
>>> if kor_score >= 80:
    print("합격입니다.")
    print("축하합니다.")
공백  들여쓰기
```

조건에 만족하면 실행하기

- if 문
  - 블록의 끝
    - 셸을 통하여 문장을 입력한 경우 빈 줄로 블록의 끝 판별
    - 별도의 에디터를 통해 작성하거나 파일에 들어 있는 경우 빈 줄이 없어도 들여쓰기가 끝나면 블록이 끝났다고 판단함

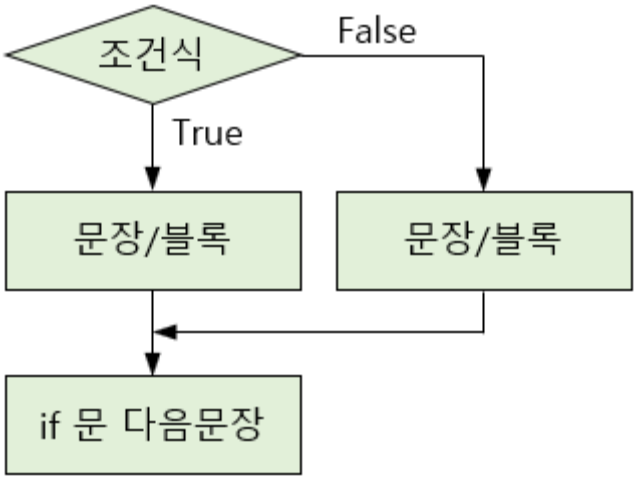
```
>>> kor_score = 90
>>> if kor_score >= 80:
        print("합격입니다.")
        print("축하합니다.")

합격입니다.
축하합니다.
>>> eng_score = 85
```

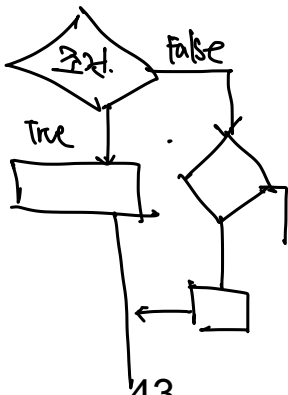
```
kor_score = 90
if kor_score >= 80:
    print("합격입니다.")
    print("축하합니다.")
eng_score = 85
```

조건에 따라 선택하기

- if-else 문
  - if 문
    - 조건식의 값이 참일 때만 문장(또는 블록)을 실행하고,
      조건식의 값이 거짓일 경우 문장(또는 블록)을 실행하지 않음
  - if-else 문
    - 조건식의 값이 참과 거짓일 경우 구분하여 실행
    - 참이나 거짓에 해당하는 부분을 반드시 한 부분은 실행

```
if 조건식:
    문장(또는 블록)
else:
    문장(또는 블록)
```

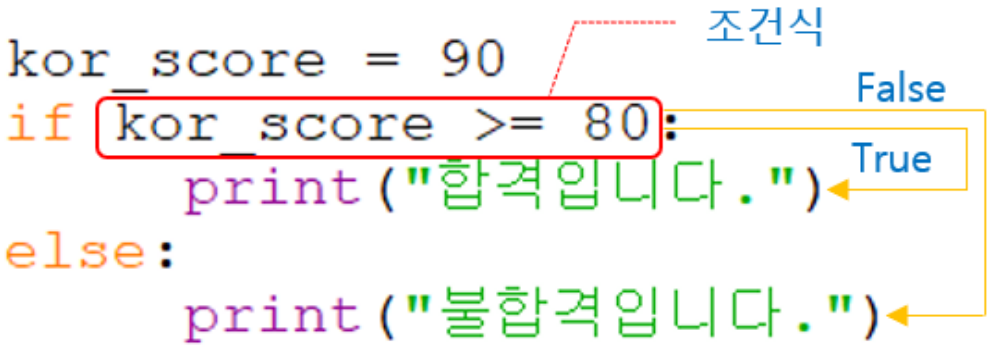




조건에 따라 선택하기

- if-else 문
  - 조건식의 값이 참이면 if 아래 문장(또는 블록) 부분 실행
  - 조건식의 값이 거짓이면 else 아래 문장(또는 블록) 부분 실행

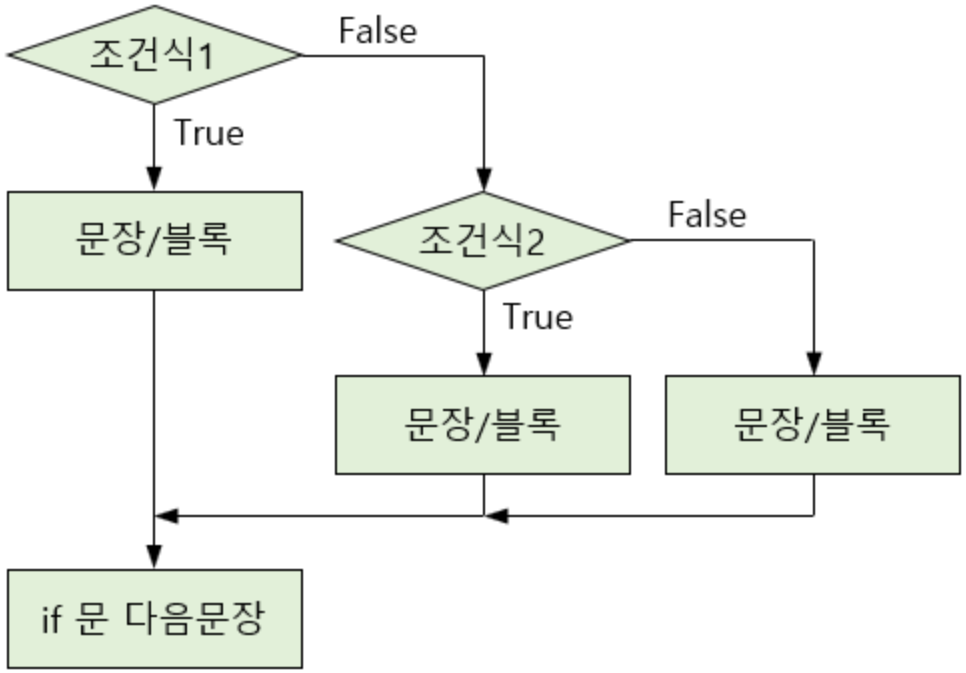
```
kor_score = 90
if kor_score >= 80:
    print("합격입니다.")
else:
    print("불합격입니다.")
```



거짓이면 다른 조건을 검사하여 선택하기

- elif 예약어, if-elif-else 문
  - if-else 문에서 조건식의 값이 거짓일 경우, else 부분에 또 다른 if 문을 추가하여 작성할 수 있음
  - 파이썬의 경우 'else if'를 합친 것과 같은 'elif' 예약어를 사용하여 추가 if 문을 작성할 수 있음

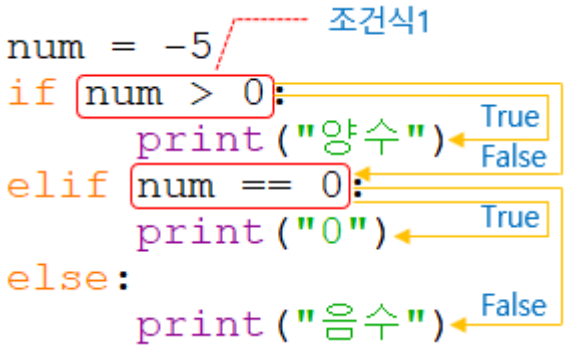
```
if 조건식1:
    문장(또는 블록)
elif 조건식2:
    문장(또는 블록)
else:
    문장(또는 블록)
```



거짓이면 다른 조건을 검사하여 선택하기

- elif 예약어, if-elif-else 문
  - 조건식1의 값이 참이면 해당 문장(또는 블록)이 실행됨, 거짓이면 다음의 조건식2를 검사함
  - 계속적으로 조건식의 값을 검사하여 참인 경우 해당 문장(또는 블록)이 실행되고, 거짓이면 다음 조건식을 검사함
  - 최종적으로 조건식의 값들이 모두 거짓인 경우 else 부분의 문장(또는 블록)이 실행됨

```
num = -5
if num > 0:
    print("양수")
elif num == 0:
    print("0")
else:
    print("음수")
```

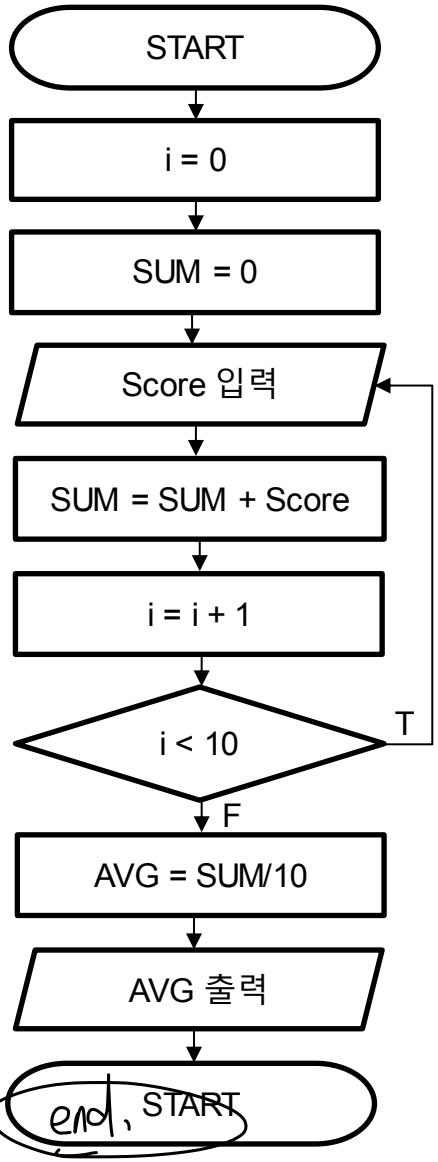


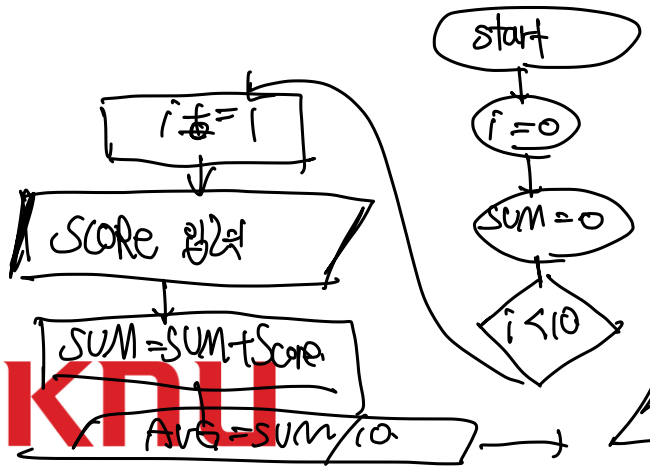
Activity

- 점수를 입력받아 90점 이상이면 A, 80점 이상이면 B, 70점 이상이면 C, 60점 이상이면 D, 60점 미만이면 F를 출력해보자.
- 위 프로그램을 수정하여 총점과 평균을 계산하고 평균이 80 이상이면 잘함, 70 이상 79 이하이면 보통, 70 미만이면 미흡을 출력해보자

순서도 복습

- 10명의 성적을 입력받아서 평균을 계산하는 알고리즘을 순서도로 나타내시오

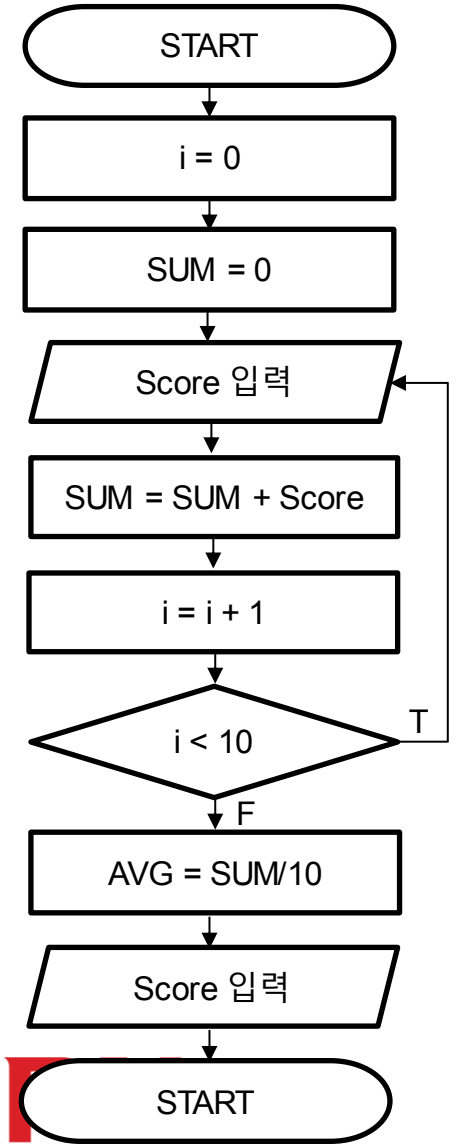




의사코드

- 자연어보다는 체계적이고 프로그래밍 언어보다는 덜 엄격한 언어로 알고리즘 표현에 주로 사용되는 코드
- 복잡한 알고리즘을 기술할 때 적합

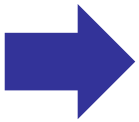
순서도를 의사코드로 변환하기




```
i = 0
sum = 0
while i < 10
        input score
        sum = sum + score
        i = i + 1
average = sum / 10
print average
```

코드 작성하기

```
i = 0
sum = 0
while i < 10
        input score
        sum = sum + score
        i = i + 1
average = sum / 10
print average
```



```
i = 0
sum = 0
while i < 10:
        score=int(input())
        sum = sum + score
        i = i + 1
average = sum / 10
print(average)
```

Activity

한국전력의 주택용 전력(저압) 요금표는 기타 계절(1.1~6.30, 9.1~12.31)에 다음과 같은 기본요금과 전력량 요금을 적용하여 계산된다(2019.12.31 기준). 전 달 전력량과 이번 달 전력량을 입력받아, 이번 달 전력량에서 전 달 전력량을 뺀 사용량을 이용하여 이번 달 전기요금을 계산하여 출력해보자. 이번 달 전력량이 전 달 전력량보다 작으면 "전력량 입력 오류"를 출력하고, 그렇지 않으면 기본요금 + 전력량 * 구간별 전력량 요금으로 전기요금을 계산한다.

구간	기본요금(원/호)	전력량 요금(원/kWh)
200kWh 이하	910	93.3
201~400kWh	1600	187.9
400kWh 초과	7300	280.6

```
전 달 전력량 : 1230
이번 달 전력량 : 1010
전력량 입력 오류
전 달 전력량 : 1010
이번 달 전력량 : 1230
이번 달 전력 사용량 = 220
전기요금 = 42938.0
```