23/03/03

1. 알고리즘이란
- 문제를 해결 하는 방법
- 문제 해결 또는 계산을 위한 프로세스나 규칙
- 주어진 입력을 원하는 출력으로 만들어주는 일련의 계산 도구

알고리즘 예제 : 숫자 맞추기 게임
문제 : A가 1부터 n사이의 숫자 하나를 마음 속으로 정한 다음, B에게 그 숫자가 무엇일지 맞춰보라고 함
게임 진행 방법
1. B가 예상 숫자를 말하면, A는 자신이 결정한 숫자와 비교해서 "같다", "크다", "작다"만 말할 수 있음
2. A가 "같다"라는 답이 나올 때까지 B는 1의 과정을 되풀이 함

B의 알고리즘(문제 해결 방법)
1번 방법.
: B는 A가 "같다"라는 답이 나올 때까지 모든 숫자를 말하는 방법 (순차 탐색)
- B의 평균 예측 횟수 = n/2
-> 시간이 오래 걸림
조금 더 쉽게 예측할 방법은 없을까?라는 고민을 하게 됨 (시간이 오래 걸린다고해서 틀린 알고리즘은 아님)

2번 방법.
B는 숫자=10을 말하여 A : "같다" -> 게임 종료 / "크다" -> 숫자 10 증가시켜 다시 말함 / "작다" -> 과정 반복
- B의 평균 예측 횟수 = n/20+4.5 [(n/10/2) + 9/2]
-> 1번 방법보다 B의 평균 예측 횟수가 빨라짐

2. 알고리즘의 표기 방법
- 자연어 : 한글 or 영어
- 프로그래밍언어 : C, C++, C#, Java 등
- **유사코드/의사코드(Pseudo-code)** : 실제 프로그램에 가깝게 계산과정을 표현할 수 있는 언어

유사코드를 사용하는 이유?
: 프로그래밍언어를 모르는 사람도 작성이 가능하고 이해하기가 쉽다. 자료를 공동으로 검토할 수 있게되어 시스템의 정확도를 향상 시킬 수 있다.

유사코드 작성시 유의사항
- (기본) 순차적으로 나열하고, 기능이 다를 때는 각각 다른 줄에 기술한다.
- (중요) 상대방이 쉽게 이해할 수 있는 방식이 가장 중요하다.
- 공백을 사용하여 기능을 쉽게 파악할 수 있도록 하는 것이 좋다.
- 마침표(.)나 컴마(,)보다는 END, IF_THEN_ELSE, END_DO와 같은 표현을 사용하는 것이 좋다.
- 무조건 분기인 GO TO 등의 사용을 피하는 것이 좋다.

의사코드 예시
1. 선택구조 예시
```
IF p THEN //if 구문 구분을 위해 대문자료 표현
    add C to D
ELSE
    add C to E
IF q THEN
    add X to Y
ELSE
    IF r THEN
    add X to Z
    ENDIF
ENDIF
```

2. 반복구조 예시
```
WHILE q DO
    S1
ENDDO
WHILE r DO
    S2
    WHILE g DO
    S3
    ENDDO
ENDDO

SUM = 0

FOR k = 100 DO
    Sum = Sum + k
ENDFOR

PRINT Sum
```
3. 최대값 구하기 예시

```
i=0

WHILE(i<n)
    A(i) = key_input
ENDWHILE

MAX = A(0)

FOR i=1 n-1 DO
    IF(MAX < A(i)) THEN
    MAX = A(i)
    ENDIF
ENDFOR

PRINT MAX
```

4. 팩토리얼 구하기 예시

```
p=1, k=1

READ n

WHILE(k<n)
    k = k + 1
    p = p * k
ENDWHILE

PRINT p
```

5-1 순차탐색 알고리즘 예시 (C-like style)

```
int seqsearch(int n, const keytype S[], keytype x)
{
    int location = 1;
    while (location <= n and S[location] ≠ x)
    ++location;
    if (location > n)
    location = 0;
    return location;
}
```

- while 루프 : 검사할 항목이 남아있고, x를 찾지 못했는가?
- if-문 : 모두 검사했지만, x를 찾지 못했는가?
- ≠ 기호를 통해 C 프로그램이 아니라 유사코드임을 알 수 있다.

5-2 순차탐색 알고리즘 예시 (Pseudo code style)

```
FUNCTION seqsearch(n, S[], x)
    location = 1
    WHILE location <= n and S[location] ≠ x
    ++location
    ENDWHILE
    IF location > n THEN
    location = 0
    ENDIF
    RETURN location
ENDFUNCTION
```

6-1. n번째 피보나치 수 구하는 예시 (순환전 방법)

- 피보나치(Fibonacci) 수열의 정의

f(0) = 0
f(1) = 1
f(n) = f(n-1) + f(n-2) / for n >= 2

- 순환적 방법 알고리즘

```
int fib (int n)
{
    if (n <= 1)
    return n;
    else
    return fib(n-1) + fib(n-2);
}
```

T(n) = 2^n/2
시간 복잡도 = 지수 복잡도(안좋음, 성능 느림)

6-2 피보나치 수 알고리즘 예시 (반복적 방법)

```
int fib2 (int n)
{
    index i;
    int f[0..n];
    f[0] = 0;
    if (n > 0)
    {
    f[1] = 1;
        for (i=2; i<=n; i++)
        f[i] = f[i-1] + f[i-2];
    }
    return f[n];
```

```
}
```

T(n) = n+1

순차적 방법보다 시간 복잡도가 줄어듬, 수행속도가 훨씬 더 빠르다.