

2과목-소프트웨어 개발

(Part 1. 데이터 입, 출력 구현)

소프트웨어 개발 총 파트

소프트웨어 개발 2과목은 총 5Part로 이루어져있다.

1장 데이터 입, 출력 구현(29.49%)

2장 통합 구현(1.92%)

3장 제품 소프트웨어 패키징(19.87%)

4장 애플리케이션 테스트 관리(35.26%)

5장 인터페이스 구현(13.46%)

데이터 입, 출력 구현

데이터 입, 출력 구현 Part는 7개의 섹션으로 구성되어 있다.

01 자료구조

02 트리(Tree)

03 정렬(Sort)

04 검색 - 이분 검색 / 해싱

05 데이터베이스 개요

06 데이터 입, 출력

07 절차형 SQL

2. 데이터 입, 출력 구현-SEC_01(자료 구조)

- 이 장을 공부하면서 반드시 알아두어야 할 키워드

; 배열, 스택, 그래프, 트리, 정렬, 데이터베이스, DBMS, 스키마, SQL, 트랜잭션

1) 자료 구조의 정의

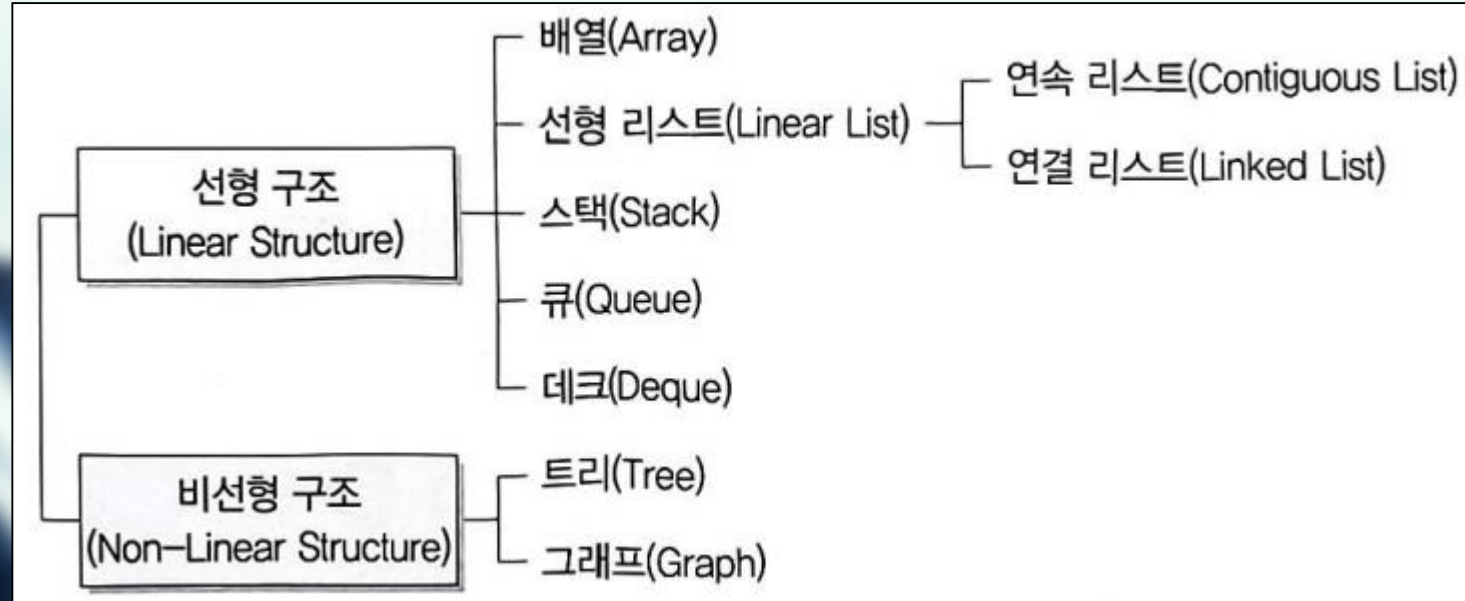
; 효율적인 프로그램을 작성할 때 가장 우선적 고려사항은 저장 공간의 효율성과 실행시간의 신속성이다.

자료 구조는 프로그램에서 사용하기 위한 자료를 기억장치의 공간 내에 저장하는 방법과 저장된 그룹 내에 존재하는 자료 간의 관계, 처리 방법 등을 연구 분석하는 것을 말한다.

- 자료 구조는 자료의 표현과 그것과 관련된 연산이다.
- 자료 구조는 일련의 자료들을 조직하고 구조화하는 것이다.
- 어떠한 자료 구조에서도 필요한 모든 연산들을 처리할 수 있다.
- 자료 구조에 따라 프로그램 실행시간이 달라진다.

2. 데이터 입, 출력 구현-SEC_01(자료 구조)

2) 자료 구조의 분류



2. 데이터 입, 출력 구현-SEC_01(자료 구조)

3) 배열(Array)

; 배열은 동일한 자료형의 데이터들이 같은 크기로 나열되어 순서를 갖고 있는 집합이다.

- 배열은 정적인 자료 구조로 기억장소의 추가가 어렵고, 데이터 삭제 시 데이터가 저장되어 있던 기억 장소는 빈 공간으로 남아있어 메모리의 낭비가 발생한다.
- 배열은 첨자(인덱스)를 이용하여 데이터에 접근한다.
- 배열은 반복적인 데이터 처리 작업에 적합한 구조이다.
- 배열은 데이터마다 동일한 이름의 변수를 사용하여 처리가 간편하다.
- 배열은 사용한 첨자(인덱스)의 개수에 따라 n차원 배열이라고 부른다.

예1) 크기가 n인 1차원 배열 arr

arr[0]	arr[1]	arr[2]	...	arr[n-1]
--------	--------	--------	-----	----------

arr은 변수의 이름이고 [숫자]는 첨자에 해당한다. 하나의 사각형은 하나의 기억장소를 가리키며, arr[0]부터 arr[n-1]까지 총 n개의 기억장소가 존재한다. 아울러 배열의 인덱스는 항상 0으로 부터 시작한다.

2. 데이터 입, 출력 구현-SEC_01(자료 구조)

3) 배열(Array)

예2) 크기가 $n * n$ 인 2차원 배열 arr

arr[0][0]	arr[0][1]	arr[0][2]	...	arr[0][n-1]
arr[1][0]	arr[1][1]	arr[1][2]	...	arr[1][n-1]
arr[2][0]	arr[2][1]	arr[2][2]	...	arr[2][n-1]
...	arr[0][n-1]
arr[n-1][0]	arr[n-1][1]	arr[n-1][2]	...	arr[n-1][n-1]

2개의 첨자가 존재하는 2차원 배열로 arr[0][0]부터 arr[n-1][n-1]까지 총 $n \times n$ 개의 기억장소가 존재한다.

2. 데이터 입, 출력 구현-SEC_01(자료 구조)

4) 선형 리스트(Linear List)

; 선형 리스트는 일정한 순서에 의해 나열된 자료 구조이다.

- 선형 리스트는 배열을 이용하는 연속 리스트(Contiguous List)와 포인터를 이용하는 연결 리스트(Linked List)로 구분된다.

- 연속 리스트(Contiguous List)

- 연속 리스트는 배열과 같이 연속되는 기억장소에 저장되는 자료 구조이다.
- 연속 리스트는 기억장소를 연속적으로 배정받기 때문에 기억장소 이용 효율은 밀도가 1로서 가장 좋다.
- 연속 리스트는 중간에 데이터를 삽입하기 위해서는 연속된 빈 공간이 있어야 하며, 삽입, 삭제 시 자료의 이동이 필요하다.

1	월
2	화
3	수
4	목
5	토
6	일
7	

삽입

금

1	월
2	화
3	수
4	목
5	금
6	토
7	일

제거

1	월
2	화
3	목
4	금
5	토
6	일
7	

포인터는 현재의 위치에서 다음 노드의 위치를 알려 주는 요소이다.

- 프론트 포인터(F. Front Pointer): 리스트를 구성 하는 최초의 노드 위치를 가리키는 요소
 - 널 포인터(Null Pointer, Nil Pointer) : 다음 노드가 없음을 나타내는 포인터로 일반적으로 마지막 노드의 링크 부분에 0, w0 등의 기호를 입력 하여 표시
- 밀도란 일정한 면적에 무엇이 뻥뻥이 들어 있는 정도를 말하는 것이다. 연속 리스트의 기억장소 이용 효율을 '밀도가 1'이라고 표현한 것은 연속 리스트는 기억장소를 연속적으로 배정받아 데이터를 기억하므로 배정된 기억장소를 빈 공간 없이 꽉 차게 사용한다는 의미이다.

2. 데이터 입, 출력 구현-SEC_01(자료 구조)

4) 선형 리스트(Linear List)

● 연결 리스트(Linked List)

- 연결 리스트는 자료들을 반드시 연속적으로 배열시키지는 않고 임의의 기억공간에 기억시키되, 자료 항목의 순서에 따라 노드의 포인터 부분을 이용하여 서로 연결시킨 자료 구조이다.
- 연결 리스트는 노드의 삽입, 삭제 작업이 용이하다.
- 기억 공간이 연속적으로 놓여 있지 않아도 저장할 수 있다.
- 연결 리스트는 연결을 위한 링크(포인터) 부분이 필요하기 때문에 순차 리스트에 비해 기억 공간의 이용 효율이 좋지 않다.
- 연결 리스트는 연결을 위한 포인터를 찾는 시간이 필요하기 때문에 접근 속도가 느리다.
- 연결 리스트는 중간 노드 연결이 끊어지면 그 다음 노드를 찾기 힘들다.

연결 리스트의 종류

단일 연결 리스트
(Singly Linked List)



이중 연결 리스트
(Doubly Linked List)



노드는 자료를 저장하는 데이터 부분과 다음 노드를 가리키는 포인터인 링크 부분으로 구성된 기억 공간이다.

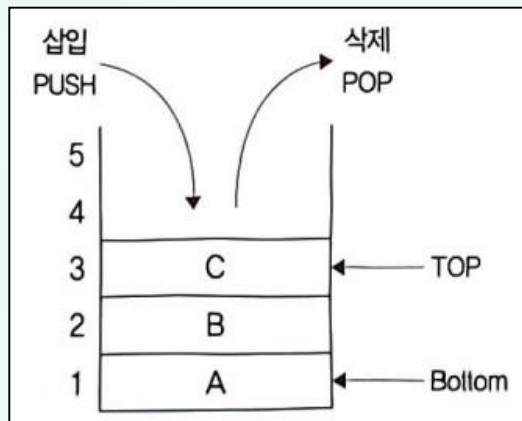
data	link
28	

2. 데이터 입, 출력 구현-SEC_01(자료 구조)

5) 스택(Stack)

; 스택은 리스트의 한쪽 끝으로만 자료의 삽입, 삭제 작업이 이루어지는 자료 구조이다.

- 스택은 가장 나중에 삽입된 자료가 가장 먼저 삭제되는 후입선출(LIFO: Last In First Out) 방식으로 자료를 처리한다.
- 스택의 응용 분야 : 함수 호출의 순서 제어, 인터럽트의 처리, 수식 계산 및 수식 표 기법, 컴파일러를 이용한 언어 번역, 부 프로그램 호출 시 복귀주소 저장, 서브루틴 호출 및 복귀 주소 저장
- 스택의 모든 기억 공간이 꽉 채워져 있는 상태에서 데이터가 삽입되면 오버플로(Overflow)가 발생하며, 더 이상 삭제할 데이터가 없는 상태에서 데이터를 삭제하면 언더플로(Underflow)가 발생한다.



2. 데이터 입, 출력 구현-SEC_01(자료 구조)

5) 스택(Stack)

- TOP : 스택으로 할당된 기억 공간에 가장 마지막으로 삽입된 자료가 기억된 위치를 가리키는 요소이다.
- Bottom: 스택의 가장 밑바닥이다.
- 자료의 삽입(Push)

Top=Top + 1	스택 포인터(Top)를 1 증가시킨다.
If Top > M Then	스택 포인터가 스택의 크기보다 크면, 더이상 자료를 삽입할 수 없으므로
Overflow	Overflow를 처리한다.
Else	그렇지 않으면
X(Top) ← Item	Item이 가지고 있는 값을 스택의 Top 위치에 삽입한다.

- M : 스택의 크기
- Top : 스택 포인터
- X: 스택의 이름
- Overflow : 스택으로 할당 받은 메모리 부분의 마지막 주소가 M번지라고 할 때, Top Pointer의 값이 M보다 커지면 스택의 모든 기억장소가 꽉 채워져 있는 상태이므로 더 이상 자료를 삽입할 수 없어 Overflow를 발생시킨다.

2. 데이터 입, 출력 구현-SEC_01(자료 구조)

5) 스택(Stack)

● 자료의 삭제(Pop)

If Top = 0 Then	스택 포인터가 0이면, 스택의 바닥이므로 더 이상 삭제할 자료가 없으므로
Underflow	Underflow를 처리한다.
Else	그렇지 않으면
Item \leftarrow X(Top)	Top 위치에 있는 값을 Item으로 옮기고
Top = Top - 1	스택 포인터를 1 감소시킨다.

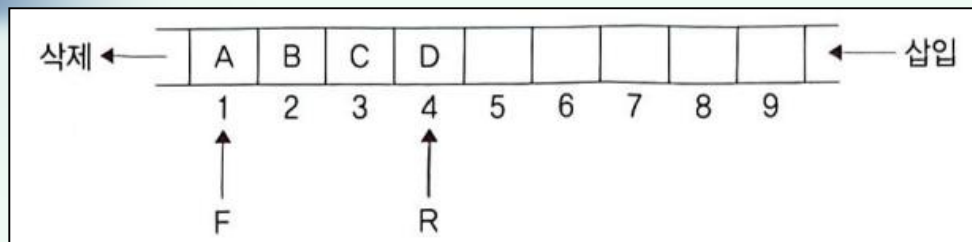
- Stack에 기억되어 있는 자료를 삭제시킬 때는 제일 먼저 삭제할 자료가 있는지 없는지부터 확인해야 한다.
- Underflow : Top Pointer가 주소 0을 가지고 있다면 스택에는 삭제할 자료가 없으므로 Underflow를 발생시킨다.

2. 데이터 입, 출력 구현-SEC_01(자료 구조)

6) 큐(Queue)

; 큐는 리스트의 한쪽에서는 삽입 작업이 이루어지고 다른 한쪽에서는 삭제 작업이 이루어지도록 구성한 자료 구조이다.

- 큐는 가장 먼저 삽입된 자료가 가장 먼저 삭제되는 선입선출(FIFO : First In First Out) 방식으로 처리한다.
- 큐는 시작과 끝을 표시하는 두 개의 포인터가 있다.



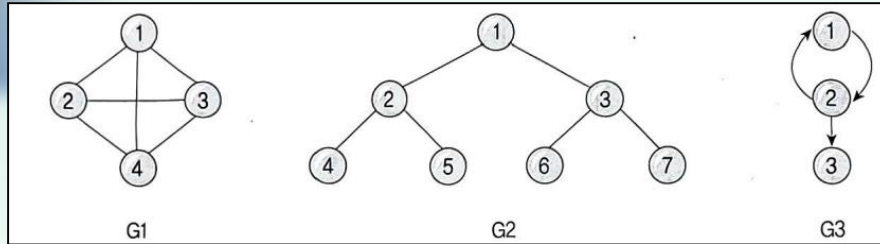
- 프론트(F, Front) 포인터 : 가장 먼저 삽입된 자료의 기억 공간을 가리키는 포인터로, 삭제 작업을 할 때 사용한다.
- 리어(R, Rear) 포인터 : 가장 마지막에 삽입된 자료가 위치한 기억 공간을 가리키는 포인터로, 삽입 작업을 할 때 사용한다.
- 큐는 운영체제의 작업 스케줄링에 사용한다.

2. 데이터 입, 출력 구현-SEC_01(자료 구조)

7) 그래프(Graph)

; 그래프 G 는 정점 $V(\text{Vertex})$ 와 간선 $E(\text{Edge})$ 의 두 집합으로 이루어진다.

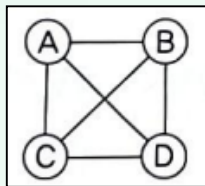
- 간선의 방향성 유무에 따라 방향 그래프와 무방향 그래프로 구분된다.
- 통신망(Network), 교통망, 이항관계, 연립 방정식, 유기화학 구조식, 무향선분 해법 등에 응용된다.
- 트리(Tree)는 사이클이 없는 그래프(Graph)이다.



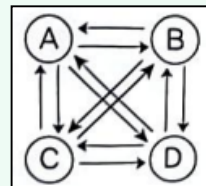
- 방향/무방향 그래프의 최대 간선 수

- n 개의 정점으로 구성된 무방향 그래프에서 최대 간선 수는 $n(n-1)/2$ 이고, 방향 그래프에서 최대 간선 수는 $n(n-1)$ 이다.

예) 정점이 4개인 경우 무방향 그래프와 방향 그래프의 최대 간선 수는 다음과 같다.



무방향 그래프의 최대 간선 수: $4(4-1)/2 = 6$



방향 그래프의 최대 간선 수 : $4(4-1) = 12$

2. 데이터 입, 출력 구현-SEC_01(자료 구조)기출 문제

기출 문제(자료 구조)

1. 다음 중 선형 구조로만 묶인 것은?

- ① 스택, 트리
- ② 큐, 데크
- ③ 큐, 그래프
- ④ 리스트, 그래프

설명

선형 구조 : 배열, 연속 리스트, 연결 리스트, 스택, 큐, 데크

비선형 구조 : 트리, 그래프

보통 큐는 선입선출 방식으로 작동한다. 반면에, 양방향 큐가 있는데 그것이 바로 데크(deque)이다. 즉, 앞, 뒤 양쪽 방향에서 엘리먼트(element)를 추가하거나 제거할 수 있다. 데크는 양 끝 엘리먼트의 추가와 삭제가 압도적으로 빠른 자료구조이다.

2. 다음은 스택의 자료 삭제 알고리즘이다. ㉠에 들어갈 내용으로 옳은 것은? (단, Top : 스택포인터, S : 스택의 이름)

3. 효율적인 프로그램을 작성할 때 가장 우선적인 고려사항은 저장 공간의 효율성과 실행시간의 신속성이다. 자료 구조의 선택은 프로그램 실행시간에 직접적인 영향을 준다. 자료 구조에 관한 설명으로 거리가 먼 것은?

- ① 자료 구조는 자료의 표현과 그것과 관련된 연산이다.
- ② 자료 구조는 일련의 자료들을 조직하고 구조화하는 것이다.
- ③ 어떠한 자료 구조에서도 필요한 모든 연산들을 처리하는 것이 가능하다.
- ④ 처리할 문제가 주어지면 평소에 주로 사용하던 자료 구조를 적용 하는 것이 좋다.

설명 : 자료 구조에 따라 프로그램 실행시간과 메모리 점유율이 달라지기 때문에 처리할 문제에 어울리는 적절한 자료구조를 선택하는 것이 옳다.

4. 다음 중 스택을 이용한 연산과 거리가 먼 것은?

- ① 선택 정렬
- ② 재귀 호출
- ③ 후위 표현(Post-Fix Expression)의 연산
- ④ 깊이 우선 탐색

2. 데이터 입, 출력 구현-SEC_01(자료 구조)기출 문제

기출 문제(자료 구조)

5. 순서가 A, B, C, D로 정해진 입력 자료를 스택에 입력한 후 출력한 결과로 불가능한 것은?

- ① D, C, B, A
- ③ C, B, A, D
- ② B, C, D, A
- ④ D, B, C, A

6. 연속 리스트의 특징이 아닌 것은?

- ① 일정한 순서에 의해 나열된 구조이다.
- ② 배열과 같이 연속되는 기억장소에 저장되는 리스트를 말한다.
- ③ 기억장소의 효율을 나타내는 메모리 밀도가 1이다.
- ④ 데이터 항목을 추가, 삭제하는 것이 용이하다.

설명 : 연속 리스트는 연속되는 기억 장소에 차례대로 데이터가 저장되는 구조이기 때문에, 중간에 데이터를 추가하거나 삭제하려면 해당 위치 이후의 모든 데이터를 이동(배열 복사)시켜야 하는 단점이 존재한다.

7. 연결 리스트(Linked List)에 대한 설명으로 거리가 먼 것은?

- ① 노드의 삽입이나 삭제가 쉽다.
- ② 노드들이 포인터로 연결되어 검색이 빠르다.
- ③ 연결을 해주는 포인터(Pointer)를 위한 추가 공간이 필요하다.
- ④ 연결 리스트 중에서 중간 노드 연결이 끊어지면 그 다음 노드를 찾기 힘들다.

설명 : 연결 리스트는 포인터로 연결되어 포인터를 찾아가는 시간이 필요하기 때문에 선형 리스트에 비해 접근 속도가 느리다.

8. 포인터를 사용하여 리스트를 나타냈을 때의 설명 중 옳지 않은 것은?

- ① 새로운 노드의 삽입이 쉽다.
- ② 기억 공간이 많이 소요된다.
- ③ 한 리스트를 여러 개의 리스트로 분리하기 쉽다.
- ④ 노드를 리스트에서 삭제하기 어렵다.

설명 : 포인터와 링크는 같은 의미를 지닌다. 링크는 다음 노드의 위치를 가리키는 주소로써, 리스트에서 특정 노드를 삭제할 때는 그 노드와 연결된 링크만 끊으면(지우면) 간단하게 삭제가 된다.

2. 데이터 입, 출력 구현-SEC_01(자료 구조)기출 문제

기출 문제(자료 구조)

9. 스택에 대한 설명으로 틀린 것은?

- ① 입출력이 한쪽 끝으로만 제한된 리스트이다.
- ② Head(front)와 Tail(rear)의 2개 포인터를 갖고 있다.
- ③ LIFO 구조이다.
- ④ 더 이상 삭제할 데이터가 없는 상태에서 데이터를 삭제하면 언더플로(Underflow)가 발생한다.

설명 : Head(front)와 Tail(rear)의 2개 포인터를 갖고 있는 자료구조는 큐(Queue)이다.

10. n개의 노드로 구성된 무방향 그래프의 최대 간선 수는?

- ① $n-1$ ② $n/2$
- ③ $n(n-1)/2$ ④ $n(n+1)$

설명 : 무방향 그래프의 최대 간선 수 구하는 공식

$n(n-1)/2$ 이며, 방향 그래프의 최대 간선 수 구하는 공식은 $n(n-1)$ 이다.

11. 자료 구조에 대한 설명으로 틀린 것은?

- ① 큐는 비선형 구조에 해당한다.
- ② 큐는 First In - First Out 처리를 수행한다.
- ③ 스택은 Last In - First Out 처리를 수행한다.
- ④ 스택은 서브루틴 호출, 인터럽트 처리, 수식 계산 및 수식 표기법에 응용된다.

설명 : 큐는 선형구조에 해당된다.

인터럽트는 CPU가 프로그램 실행하고 있을 때, 입출력 하드웨어 등의 장치나 예외상황이 발생하여 처리가 필요할 경우에 마이크로프로세서에게 알려 처리할 수 있도록 하는 것을 의미한다.

인터럽트의 종류는 크게 하드웨어 인터럽트가 있고, 소프트웨어 인터럽트로 나뉜다.

12. 스택(STACK)의 응용분야로 거리가 먼 것은?

- ① 인터럽트의 처리
- ② 수식의 계산
- ③ 서브루틴의 복귀 번지 저장
- ④ 운영체제의 작업 스케줄링

설명 : 스택은 가장 나중에 의뢰된 작업이 가장 먼저 처리되는 후입선출

2. 데이터 입, 출력 구현-SEC_01(자료 구조)기출 문제

기출 문제(자료 구조)

13. 스택(Stack)에 대한 옳은 내용으로만 나열된 것은?

- ㉠ FIFO 방식으로 처리된다.
- ㉡ 순서 리스트의 뒤(Rear)에서 노드가 삽입되며, 앞(Front)에서 노드가 제거된다.
- ㉢ 선형 리스트의 양쪽 끝에서 삽입과 삭제가 모두 가능한 자료 구조이다.
- ㉣ 인터럽트 처리, 서브루틴 호출 작업 등에 응용된다.

① ㉠, ㉡

② ㉡, ㉣

③ ㉢

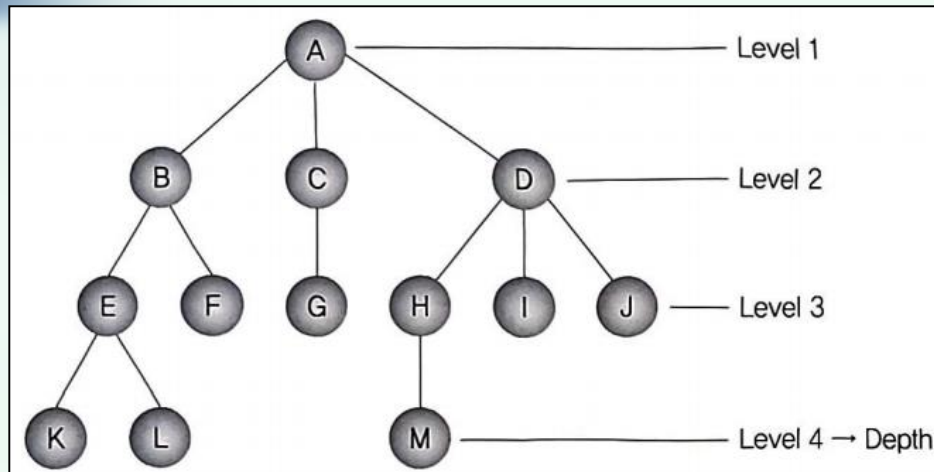
④ ㉠, ㉡, ㉣, ㉣

2. 데이터 입, 출력 구현-SEC_02(트리(Tree))

1) 트리의 개요

; 트리는 정점(Node, 노드)과 선분(Branch, 가지)을 이용하여 사이클을 이루지 않도록 구성된 그래프(Graph)의 특수한 형태이다.

- 트리는 하나의 기억 공간을 노드(Node)라고 하며, 노드와 노드를 연결하는 선을 링크라고 한다.
- 트리는 가족의 계보(족보), 조직도 등을 표현하기에 적합하다.
- 트리 관련 용어



노드(Node) : 트리의 기본 요소로서 자료 항목과 다른 항목에 대한 가지(Branch)를 합친 것

예) A, B, C, D, E, F, G, H, I, J, K, L, M

근 노드(Root Node): 트리의 맨 위에 있는 노드

예) A

디그리(Degree, 차수) : 각 노드에서 뻗어 나온 가지의 수

예) A = 3, B = 2, C = 1, D = 3

단말 노드(Terminal Node) = 잎 노드(Leaf Node) : 자식이 하나도 없는 노드, 즉 디그리가 0인 노드

예) K, L, F, G, M, I, J

자식 노드(Son Node) : 어떤 노드에 연결된 다음 레벨의 노드들

예) D의 자식 노드: H, I, J

부모 노드(Parent Node) : 어떤 노드에 연결된 이전 레벨의 노드들

예) E, F의 부모 노드 : B

형제 노드(Brother Node, Sibling) : 동일한 부모를 갖는 노드들

예) H의 형제 노드 : I, J

트리의 디그리 : 노드들의 디그리 중에서 가장 많은 수

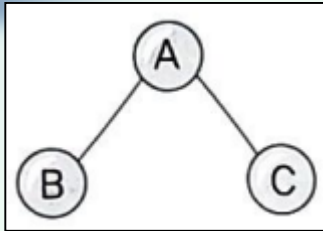
예) 노드 A나 D가 3개의 디그리를 가지므로 앞 트리의 디그리는 3이다.

2. 데이터 입, 출력 구현-SEC_02(트리(Tree))

2) 트리의 운행법

; 트리를 구성하는 각 노드들을 찾아가는 방법을 운행법(Traversal)이라 한다.

- 이진 트리를 운행하는 방법은 산술식의 표기법과 연관성을 갖는다.
- 이진 트리의 운행법은 다음 세 가지가 있다.
- 이진 트리 운행법의 이름은 Root의 위치가 어디 있느냐에 따라 정해진 것이다. 즉 Root가 앞(Pre)에 있으면 Preorder, (In)에 있으면 Inorder, 뒤(Post)에 있으면 Postorder다.

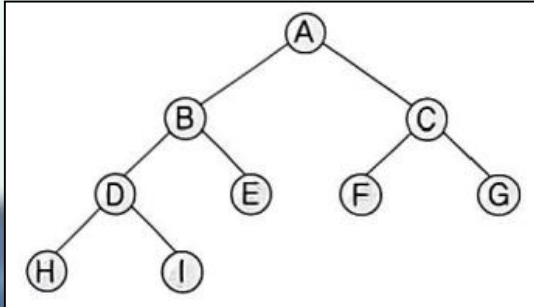


- Preorder : Root -> Left -> Right 순으로 운행한다. A, B, C
- Inorder : Left -> Root -> Right 순으로 운행한다. B, A, C
- Postorder 운행 : Left -> Right -> Root 순으로 운행한다. B, C, A

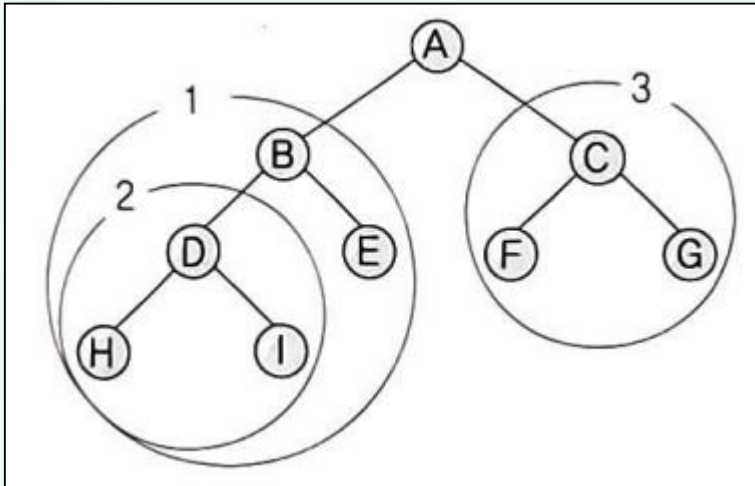
2. 데이터 입, 출력 구현-SEC_02(트리(Tree))

2) 트리의 운행법

예제) 다음 트리를 Inorder, Preorder, Postorder 방법으로 운행했을 때 각 노드를 방문한 순서는?



● Preorder 운행법의 방문 순서



※ 서브트리를 하나의 노드로 생각할 수 있도록 그림과 같이 서브 트리 단위로 묶는다. Preorder, Inorder, Postorder 모두 공통으로 사용한다.

① Preorder는 Root -> Left -> Right 이므로 A13이 된다.

② 1은 B2E이므로 AB2E3이 된다.

③ 2는 DHI이므로 ABDHIE3이 된다.

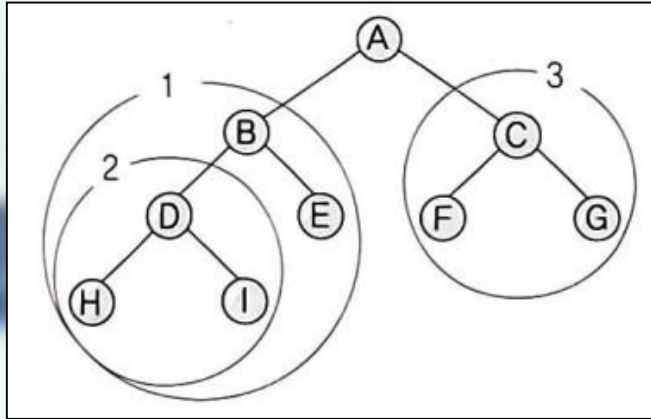
④ 3은 CFG이므로 ABDHIECFG가 된다.

방문 순서: ABDHIECFG

2. 데이터 입, 출력 구현-SEC_02(트리(Tree))

2) 트리의 운행법

● Inorder 운행법의 방문 순서



- ① Inorder는 Left -> Root -> Right이므로 1A3이 된다.
 - ② 1은 2BE이므로 2BEA3이 된다.
 - ③ 2는 HDI이므로 HDIBEA3이 된다.
 - ④ 3은 FCG이므로 HDIBEAFCG가 된다.
- 방문 순서 : HDIBEAFCG

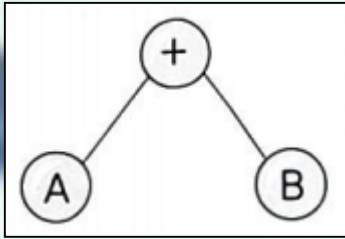
● Postorder 운행법의 방문 순서

- ① Postorder는 Left -> Right -> Root이므로 13A가 된다.
 - ② 1은 2EB이므로 2EB3A가 된다.
 - ③ 2는 HID이므로 HIDEB3A가 된다.
 - ④ 3은 FGC이므로 HIDEBFGCA가 된다.
- 방문 순서 : HIDEBFGCA

2. 데이터 입, 출력 구현-SEC_02(트리(Tree))

3) 수식의 표기법

; 산술식을 계산하기 위해 기억공간에 기억시키는 방법으로 이진 트리를 많이 사용한다. 이진 트리로 만들어진 수식을 인오더, 프리오더, 포스트오더로 운행하면 각각 중위(Infix), 전위(Prefix), 후위(Postfix) 표기법이 된다.



- 전위 표기법(Prefix) : 연산자 -> Left -> Right, +AB
- 중위 표기법(InFix) : Left -> 연산자 -> Right, A+B
- 후위 표기법(PostFix) : Left -> Right -> 연산자, AB+

2. 데이터 입, 출력 구현-SEC_02(트리(Tree))

3) 수식의 표기법

; Infix 표기를 Postfix Prefix로 바꾸기

● Postfix나 Prefix는 스택을 이용하여 처리하므로 Infix는 Postfix나 Prefix로 바꾸어 처리한다.

- 예제1) 다음과 같이 Infix로 표기된 수식을 Prefix와 Postfix로 변환하시오.

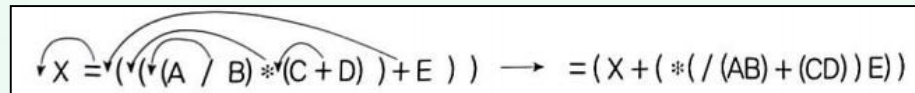
$$X = A / B * (C + D) + E$$

Prefix로 변환하기

① 연산 우선순위에 따라 괄호로 묶는다.

$$(X = (((A / B) * (C + D)) + E))$$

② 연산자를 해당 괄호의 앞(왼쪽)으로 옮긴다.


$$\rightarrow = (X + (* (/ (AB) + (CD)) E))$$

③ 필요 없는 괄호를 제거한다.

prefix 표기 : =X+*/AB+CDE

2. 데이터 입, 출력 구현-SEC_02(트리(Tree))

3) 수식의 표기법

; Infix 표기를 Postfix Prefix로 바꾸기

● Postfix나 Prefix는 스택을 이용하여 처리하므로 Infix는 Postfix나 Prefix로 바꾸어 처리한다.

- 예제1) 다음과 같이 Infix로 표기된 수식을 Prefix와 Postfix로 변환하시오.

$$X = A / B * (C + D) + E$$

Postfix로 변환하기

① 연산 우선순위에 따라 괄호로 묶는다.

$$(X = (((A / B) * (C + D)) + E))$$

② 연산자를 해당 괄호의 뒤(오른쪽)로 옮긴다.

$$(X = (((A / B)^* (C + D)^+ + E)^+)^+)^+ \rightarrow (X((A B)/(C D)+)*E+=)$$

③ 필요 없는 괄호를 제거한다.

Postfix 표기: $XAB/CD+*E+=$

2. 데이터 입, 출력 구현-SEC_02(트리(Tree))

3) 수식의 표기법

; Postfix나 Prefix로 표기된 수식을 Infix로 바꾸기

- 예제2) 다음과 같이 Postfix로 표기된 수식을 Infix로 변환하시오.

A B C - / D E F + * +

Postfix는 Infix 표기법에서 연산자를 해당 피연산자 두 개의 뒤로 이동한 것이므로 연산자를 다시 해당 피연산자 두 개의 가운데로 옮기면 된다.

① 먼저 인접한 피연산자 두 개와 오른쪽의 연산자를 괄호로 묶는다.

$((A (B C -) /) (D (E F +) *) +)$

② 연산자를 해당 피연산자의 가운데로 이동시킨다.

$((A (B C -) /) (D (E F +) *) +) \rightarrow ((A / (B - C)) + (D * (E + F)))$

③ 필요 없는 괄호를 제거한다.

$((A / (B - C)) + (D * (E + F))) \rightarrow A / (B - C) + D * (E + F)$

2. 데이터 입, 출력 구현-SEC_02(트리(Tree))

3) 수식의 표기법

; Postfix나 Prefix로 표기된 수식을 Infix로 바꾸기

- 예제3) 다음과 같이 Prefix로 표기된 수식을 Infix로 변환하시오.

+ / A - B C * D + E F

Prefix는 Infix 표기법에서 연산자를 해당 피연산자 두 개의 앞으로 이동한 것이므로 연산자를 다시 해당 피연산자 두 개의 가운데로 옮기면 된다.

① 먼저 인접한 피연산자 두 개와 오른쪽의 연산자를 괄호로 묶는다.

$(+ (/ A (- B C)) (* D (+ E F)))$

② 연산자를 해당 피연산자의 가운데로 이동시킨다.

$(+ (/ A (- B C)) (* D (+ E F))) \rightarrow ((A/(B-C)) + (D * (E+F)))$

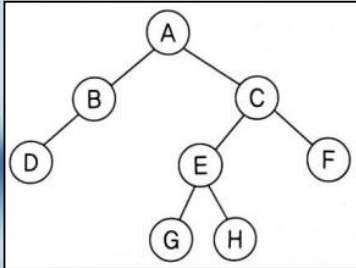
③ 필요 없는 괄호를 제거한다.

$((A/(B-C)) + (D*(E+F))) \rightarrow A / (B - C) + D * (E+F)$

2. 데이터 입, 출력 구현-SEC_02(트리(Tree)) 기출 및 예상 문제

기출 문제 및 예상 문제(트리(Tree))

1. 다음 트리의 차수(Degree)와 단말 노드(Terminal Node)의 수는?



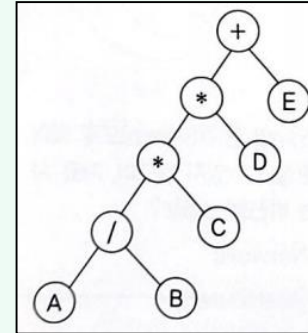
- ① 차수 : 4, 단말 노드 : 4
- ② 차수 : 2, 단말 노드 : 4
- ③ 차수 : 4, 단말 노드 : 8
- ④ 차수 : 2, 단말 노드 : 8

설명 : 트리의 차수(Degree)는 가장 차수가 많은 노드의 차수이고 단말 노드(잎 노드)는 자식이 하나도 없는 노드를 의미한다.

2. 트리 구조에 대한 용어 설명 중 옳지 않은 것은?

- ① 어떤 노드의 서브트리 수를 그 노드의 차수라고 한다.
- ② 차수가 0인 노드를 단말노드라고 한다.
- ③ 같은 부모 노드를 가지는 노드를 형제 노드라고 한다.

3. 다음 트리를 전위 순회(Preorder Traversal)한 결과는?



- ① + * A B / * C D E
- ② A B / C * D * E +
- ③ A / B * C * D + E
- ④ + * * / A B C D E

설명 : Preorder는 Root -> Left -> Right 이다.

먼저 서브트리를 하나의 노드로 생각할 수 있도록 서브트리 단위로 묶는다.

1. Preorder는 Root -> Left -> Right 이므로 +1E 가 된다.

2. 1은 *2D이므로 +*2DE 가 된다

3. 2는 *3C이므로 +**3CDE 가 된다.

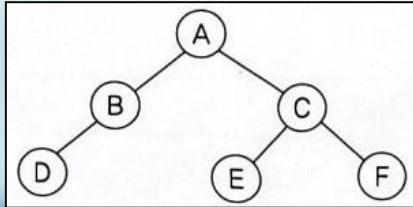
4. 3은 /AB이므로 +**/ABCDE 가 된다.

4. 다음 트리를 Preorder 운행법으로 운행할 경우 가장 먼저 탐색되는

2. 데이터 입, 출력 구현-SEC_02(트리(Tree)) 기출 및 예상 문제

기출 문제 및 예상 문제(트리(Tree))

5. 다음 트리에 대한 INORDER 운행 결과는?



- ① D B A E C F
- ② A B D C E F
- ③ D B E C F A
- ④ A B C D E F

설명 : Inorder는 Left -> Root -> Right 순이다. 1A2가 된다

1. 1은 DB이다. DBA2가 된다.

2. 2는 ECF이다. DBAECF가 된다.

6. 다음 Postfix 연산식에 대한 연산 결과로 옳은 것은?

$$3\ 4\ *\ 5\ 6\ *\ +$$

- ① 35
- ② 42
- ③ 77
- ④ 360

설명 : 후위 표기법(Postfix)이란 연산자가 해당 피연산자 2개의 뒤 (오른쪽)에 표기되어 있는 것을 말한다. 그러므로 피연산자 2개와 연산자를 묶은 후 연산자를 피연산자 사이에

7. 다음 전위식(Prefix)을 후위식(Postfix)으로 옳게 표현한 것은?

$$- / * A + B C D E$$

- ① A B C + D / * E -
- ② A B * C D / + E -
- ③ A B * C + D / E -
- ④ A B C + * D / E -

설명 : 인접한 피연산자 두 개와 왼쪽의 연산자를 괄호로 묶고 해당 괄호의 뒤(오른쪽)로 옮긴다.

$(- (/ (* A (+ B C))) D) E \rightarrow A B C + * D / E -$ 이 된다.

8. 다음과 같이 주어진 후위 표기 방식의 수식을 중위 표기 방식으로 나타낸 것은?

$$A\ B\ C\ -\ /\ D\ E\ F\ +\ *\ +$$

- ① A / (B - C) + F * E + D
- ② A / (B - C) + D * (E + F)
- ③ A / (B - C) + D + E * F
- ④ A / (B - C) * D + E + F

설명 : Postfix는 Infix로 표기된 것에서 연산자를 해당 피연산자 2개

2. 데이터 입, 출력 구현-SEC_02(트리(Tree)) 기출 및 예상 문제

기출 문제 및 예상 문제(트리(Tree))

9. 다음의 중위식(Infix)을 전위(Prefix)식으로 옳게 변환한 것은?

$$A * B + C - D / E$$

① $- + * A B C / D E$

② $A B * C + D E / -$

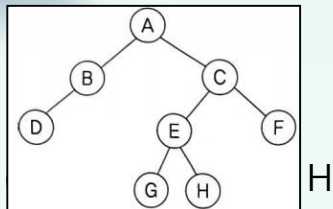
③ $A B C D E * + - /$

④ $* + - / A B C D E$

설명 : 전위(Prefix)식은 두 개의 피연산자의 앞쪽(왼쪽)에 연산자를 표식한다.

$$(((A * B) + C) - (D / E)) = - + * A B C / D E$$

10. 아래 이진 트리를 후위 순서(Postorder)로 운행한 결과는?



② $D B G H E F C A$

③ $A B D C E G H F$

11. 그래프의 특수한 형태로, 노드(Node)와 선분(Branch)으로 되어 있고, 정점 사이에 사이클(Cycle)이 형성되어 있지 않으며, 자료 사이의 관계성이 계층 형식으로 나타나는 비선형 구조는?

① Tree

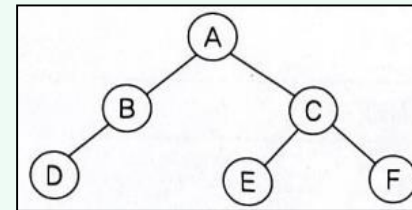
② Network

③ Stack

④ Queue

설명 : 비선형 구조는 Tree와 Graph 밖에 없다.

12. 아래의 트리에서 트리의 깊이는 얼마인가?



① 1

② 2

③ 3

④ 4

설명 : 트리의 깊이(Depth)는 곧 루트노드를 1로 보고 따라가서 마지막에 있는 값이 n이 된다. 그것은 바로 트리의 깊이이다.

2. 데이터 입, 출력 구현-SEC_03(정렬(Sort))

1) 삽입 정렬(Insertion Sort)

; 삽입 정렬은 가장 간단한 정렬 방식으로 이미 순서화된 파일에 새로운 하나의 레코드를 순서에 맞게 삽입시켜 정렬한다.

- 두 번째 키와 첫 번째 키를 비교해 순서대로 나열(1회전)하고, 이어서 세 번째 키를 첫 번째, 두 번째 키와 비교해 순서대로 나열(2회전)하고, 계속해서 n번째 키를 앞의 n-1개의 키와 비교하여 알맞은 순서에 삽입하여 정렬하는 방식이다.
- 평균과 최악 모두 수행 시간 복잡도는 $O(n^2)$ 이다.
- 시간 복잡도의 가장 간단한 정의는 알고리즘의 성능을 설명하는 것이다. 다른 의미로는 알고리즘을 수행하기 위해 프로세스가 수행해야 하는 연산을 수치화 한 것이다. 왜 실행시간이 아닌 연산수치로 판별할까? 명령어의 실행시간은 컴퓨터의 하드웨어 또는 프로그래밍 언어에 따라 편차가 크게 달라지기 때문에 명령어의 실행 횟수만을 고려하는 것이다.

$$O(1) > O(\log N) > O(N) > O(N \log N) > O(N^2) > O(2^N) > O(N!)$$

- 삽입 정렬의 장단점
 - 최선의 경우 $O(N)$ 이라는 엄청나게 빠른 효율성을 가지고 있다.
 - 성능이 좋아서 다른 정렬 알고리즘의 일부로 사용될 만큼 좋은 정렬법이다.
 - 최악의 경우 $O(n^2)$ 이라는 시간 복잡도를 갖게 되며, 데이터의 크기, 종류에 따라 실행 편차가 크다.

2. 데이터 입, 출력 구현-SEC_03(정렬(Sort))

1) 삽입 정렬(Insertion Sort)

; 예제) 8, 5, 6, 2, 4를 삽입 정렬로 정렬하시오.

초기 상태 :

8	5	6	2	4
---	---	---	---	---

1회전 :

8	5	6	2	4
---	---	---	---	---

5	8	6	2	4
---	---	---	---	---

두 번째 값을 첫 번째 값과 비교하여 5를 첫 번째 자리에 삽입하고 8을 한 칸 뒤로 이동시킨다.

2회전 :

5	8	6	2	4
---	---	---	---	---

5	6	8	2	4
---	---	---	---	---

세 번째 값을 첫 번째, 두 번째 값과 비교하여 6, 8자리에 삽입하고 8은 한 칸 뒤로 이동시킨다.

3회전 :

5	6	8	2	4
---	---	---	---	---

2	5	6	8	4
---	---	---	---	---

네 번째 값 2를 처음부터 비교하여 맨 처음에 삽입하고 나머지를 한 칸씩 뒤로 이동시킨다.

4회전 :

2	5	6	8	4
---	---	---	---	---

2	4	5	6	8
---	---	---	---	---

다섯 번째 값 4를 처음부터 비교하여 5자리에 삽입하고 나머지를 한 칸씩 뒤로 이동시킨다.

2. 데이터 입, 출력 구현-SEC_03(정렬(Sort))

2) 셸 정렬(Shell Sort)

; 셸 정렬은 삽입 정렬(Insertion Sort)을 확장한 개념이다.

- 입력 파일을 어떤 매개변수(b)의 값으로 서브 파일을 구성하고, 각 서브 파일을 Insertion 정렬 방식으로 순서 배열하는 과정을 반복하는 정렬 방식($h = \sqrt[3]{n}$), 즉 임의의 레코드 키와 h 값 만큼 떨어진 곳의 레코드 키를 비교하여 순서화되어 있지 않으면 서로 교환하는 것을 반복하는 정렬 방식이다.
- 입력 파일이 부분적으로 정렬되어 있는 경우에 유리한 방식이다.
- 평균 수행 시간 복잡도는 $O(n^{1.5})$ 이고, 최악의 수행 시간복잡도는 $O(n^2)$ 이다.
- 셸 정렬의 장단점
 - 삽입 정렬의 단점을 보완해서 만든 정렬법으로 삽입 정렬도 성능이 뛰어난 편이지만 더 뛰어난 성능을 갖는 정렬법이다.
 - 일정한 간격에 따라서 배열을 바라봐야 한다. 즉, 이 '간격'을 잘못 설정한다면 성능이 굉장히 안 좋아질 수 있다.

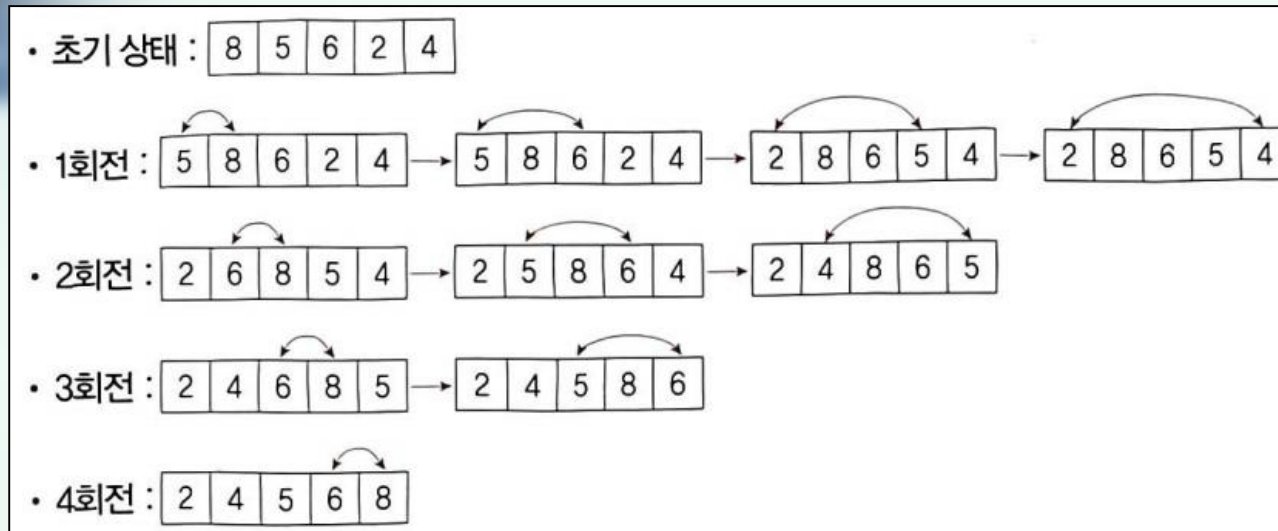
2. 데이터 입, 출력 구현-SEC_03(정렬(Sort))

3) 선택 정렬(Selection Sort)

; 선택 정렬은 n 개의 레코드 중에서 최소값을 찾아 첫 번째 레코드 위치에 놓고, 나머지 $(n-1)$ 개 중에서 다시 최소값을 찾아 두 번째 레코드 위치에 놓는 방식을 반복하여 정렬하는 방식이다.

● 평균과 최악 모두 수행 시간복잡도는 $O(n^2)$ 로써 그리 썩 좋은 알고리즘은 아니다.

예제) 8, 5, 6, 2, 4 를 선택 정렬로 정렬하시오.



● 선택 정렬의 장단점

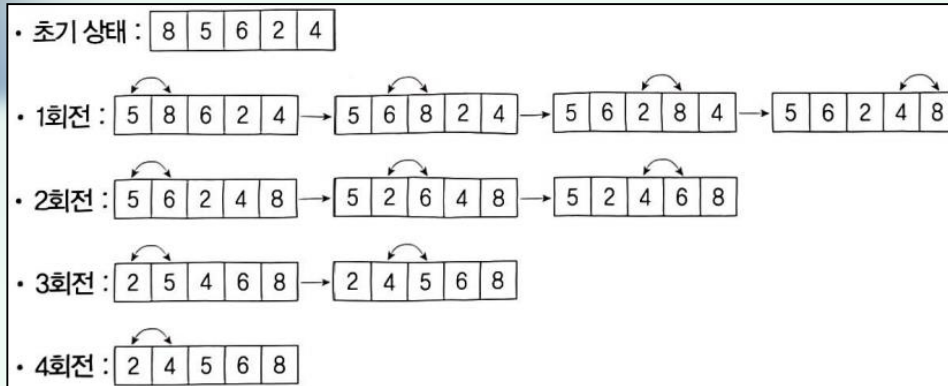
- 선택 정렬은 또한 버블 정렬과 마찬가지로 구현이 쉬운 편에 속하는 정렬법이다.
- 정렬을 위한 비교 횟수는 많지만 실제로 교환하는 횟수는 적기 때문에 많은 교환이 일어나야 하는 자료 상태에서 효율적으로 사용될 수 있다.

2. 데이터 입, 출력 구현-SEC_03(정렬(Sort))

4) 버블 정렬(Bubble Sort)

- 버블 정렬은 주어진 파일에서 인접한 두 개의 레코드 키 값을 비교하여 그 크기에 따라 레코드 위치를 서로 교환하는 정렬 방식이다.
- 평균과 최악 모두 수행 시간 복잡도는 $O(n^2)$ 로써 그리 썩 좋은 알고리즘은 아니다.

예제) 8, 5, 6, 2, 4를 버블 정렬로 정렬하시오.



- 버블 정렬의 장단점
 - 일단 구현이 쉽다. Bubble정렬은 인접한 값만 계속해서 비교하면 되는 방식으로 굉장히 구현이 쉬운 편이다. 코드 자체가 직관적이다.
 - 굉장히 비효율적이다. 최악이든 최선이든 $O(n^2)$ 이라는 시간복잡도를 갖기 때문에 사실 알고리즘에서 효율적인 정렬방법으로 사용되지는 않는다.

2. 데이터 입, 출력 구현-SEC_03(정렬(Sort))

5) 퀵 정렬(Quick Sort)

; 퀵 정렬은 레코드의 많은 자료 이동을 없애고 하나의 파일을 부분적으로 나누어 가면서 정렬하는 방법으로 키를 기준으로 작은 값은 왼쪽에, 큰 값은 오른쪽 서브 파일로 분해시키는 방식으로 정렬한다.

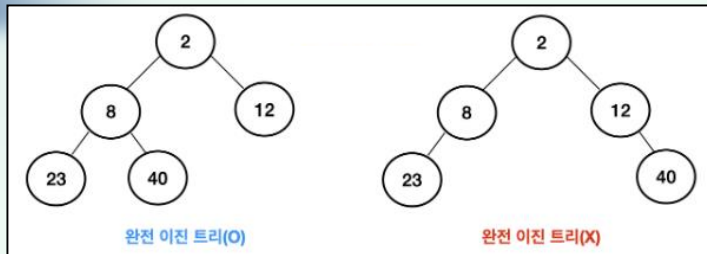
- 위치에 관계 없이 임의의 키를 분할 원소로 사용할 수 있다.
- 정렬 방식 중에서 가장 빠른 방식이다.
- 프로그램에서 되부름을 이용하기 때문에 스택(Stack)이 필요하다.
- 분할(Divide)과 정복(Conquer)을 통해 자료를 정렬한다.
 - 분할(Divide) : 기준 값인 피벗(Pivot)을 중심으로 정렬할 자료들을 2개의 부분 집합으로 나눈다.
 - 정복(Conquer) : 부분 집합의 원소들 중 피벗(Pivot)보다 작은 원소들은 왼쪽, 피벗(Pivot)보다 큰 원소들은 오른쪽 부분 집합으로 정렬한다.
 - 부분 집합의 크기가 더 이상 나누어질 수 없을 때까지 분할과 정복을 반복 수행한다.
- 퀵 정렬의 장단점
 - 기준 값에 의한 분할을 통해서 구현하는 정렬법으로써, 분할 과정에서 $\log N$ 이라는 시간이 걸리게 되고 전체적으로 보게 되면 $N \log N$ 으로써 실행시간이 빠른 편이다.
 - 기준값(Pivot)에 따라서 시간 복잡도가 크게 달라진다.

2. 데이터 입, 출력 구현-SEC_03(정렬(Sort))

6) 힙 정렬(Heap Sort)

; 힙 정렬은 전이진(완전 이진) 트리(Complete Binary Tree)를 이용한 정렬 방식이다.

- 구성된 전이진 트리를 Heap Tree로 변환하여 정렬한다.
- 전이진 트리란 각 노드가 최대 2개의 자식 노드를 갖는 트리 형태의 자료구조로서 마지막 레벨을 제외한 모든 노드는 완전히 채워져 있어야 한다. 또한, 최 하단 레벨의 노드는 좌측만 노드가 채워져 있거나 좌측과 우측 모두 채워져 있어야 한다.



- 힙 정렬의 장단점

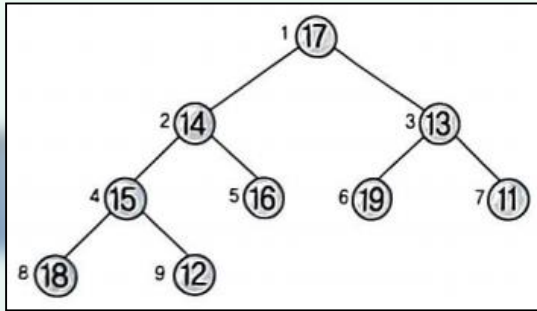
- 추가적인 메모리를 필요로 하지 않으면서 항상 $O(N\log N)$ 이라는 시간복잡도를 가지는 굉장히 정렬법들 중에서 효율적인 정렬법이라고 볼 수 있다. 퀵 정렬도 효율적이라고 볼 수 있지만 최악의 경우 시간이 오래 걸린다는 단점이 있지만 힙 정렬의 경우 항상 $O(N\log N)$ 으로 보장된다는 장점이 있다.
- 데이터의 상태에 따라서 다른 정렬법들에 비해서 조금 느린 편이다. 또한, 안정성(Stable)을 보장받지 못한다는 단점이 있다.

2. 데이터 입, 출력 구현-SEC_03(정렬(Sort))

6) 힙 정렬(Heap Sort)

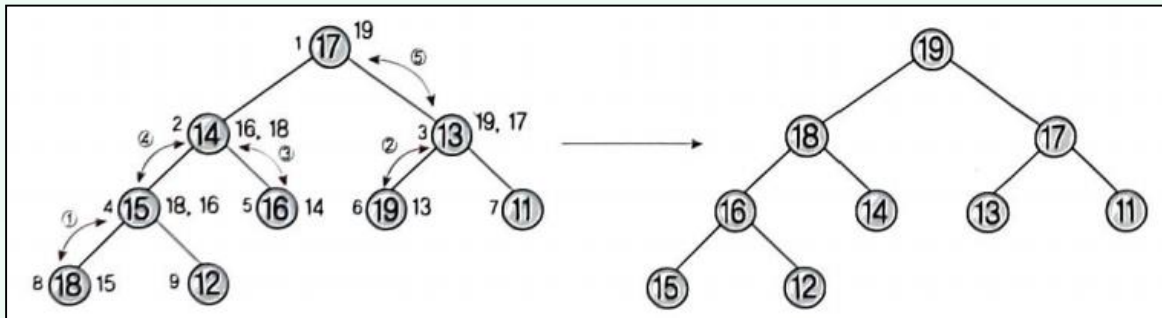
예제) 17, 14, 13, 15, 16, 19, 11, 18, 12를 Heap 트리로 구성하시오.

① 주어진 파일의 레코드들을 전이진 트리로 구성한다.



② 전이진 트리의 노드의 역순으로 자식 노드와 부모 노드를 비교하여 큰 값을 위로 올린다.

③ 교환된 노드들을 다시 검토하여 위의 과정을 반복한다.



2. 데이터 입, 출력 구현-SEC_03(정렬(Sort))

7) 2-Way 합병 정렬(Merge Sort)

; 2-Way Merge Sort는 이미 정렬되어 있는 두 개의 파일을 한 개의 파일로 합병하는 정렬 방식이다.

그 방법은 다음과 같다.

- 두 개의 키들을 한 쌍으로 하여 각 쌍에 대하여 순서를 정한다.
- 순서대로 정렬된 각 쌍의 키들을 합병하여 하나의 정렬된 서브리스트로 만든다.
- 위 과정에서 정렬된 서브리스트들을 하나의 정렬된 파일이 될 때까지 반복한다.
- 평균과 최악 모두 시간복잡도는 $O(N\log N)$ 이다.

예제) 71, 2, 38, 5, 7, 61, 11, 26, 53, 42를 2-Way 합병 정렬로 정렬하시오.

- 1회전 : 두 개씩 묶은 후 각각의 묶음 안에서 정렬한다.

(71, 2) (38, 5) (7, 61) (11, 26) (53, 42)

(2, 71) (5, 38) (7, 61) (11, 26) (42, 53)

- 2회전 : 묶여진 묶음을 두 개씩 묶은 후 각각의 묶음 안에서 정렬한다.

((2, 71) (5, 38)) ((7, 61) (11, 26)) (42, 53)

(2, 5, 38, 71) (7, 11, 26, 61) (42, 53)

2. 데이터 입, 출력 구현-SEC_03(정렬(Sort))

7) 2-Way 합병 정렬(Merge Sort)

예제) 71, 2, 38, 5, 7, 61, 11, 26, 53, 42를 2-Way 합병 정렬로 정렬하시오.

- 3회전 : 묶여진 묶음을 두 개씩 묶은 후 각각의 묶음 안에서 정렬한다.

((2, 5, 38, 71) (7, 11, 26, 61)) (42, 53)

(2, 5, 7, 11, 26, 38, 61, 71) (42, 53)

- 4회전 : 묶여진 묶음 두 개를 하나로 묶은 후 정렬한다.

((2, 5, 7, 11, 26, 38, 61, 71) (42, 53))

2, 5, 7, 11, 26, 38, 42, 53, 61, 71

● 2-Way 합병 정렬의 장단점

- 퀵 정렬과 비슷하게 원본 배열을 반씩 분할해 가면서 정렬하는 정렬법으로써 분할 하는 과정에서 $\log N$ 만큼의 시간이 걸린다. 즉, 최종적으로 보게 되면 $N \log N$ 이 된다. 퀵 정렬과 달리, Pivot을 설정하거나 그런 과정 없이 무조건 절반으로 분할하기 때문에 Pivot에 따라서 성능이 안 좋아지거나 하는 경우가 없다. 따라서 항상 $O(N \log N)$ 이라는 시간 복잡도를 갖게 된다.
- 장점만 본다면 퀵 정렬보다는 무조건 병합 정렬을 사용하는 것이 좋다고 생각할 수 있지만 가장 큰 단점은 '추가적인 메모리 필요'이다. 병합정렬은 임시배열에 원본 맵을 계속해서 옮겨주면서 정렬을 하는 방법이기 때문이다.

2. 데이터 입, 출력 구현-SEC_03(정렬(Sort))

8) 기수 정렬(Radix Sort) = Bucket Sort

; 정렬되지 않은 자료를 버킷이라는 단위 기억 장소에 여러 그룹으로 정렬하고 버킷별 키 값에 따라 다시 정렬하는 알고리즘이다.

- 정렬법들 중에서 $O(N)$ 이라는 말도 안 되는 시간 복잡도를 갖는 정렬법으로써 일단 엄청나게 빠르다.
- 정렬법에서 $O(N\log N)$ 을 깰 수 있는 방법은 없다고 알려져 있는데, 그 방법을 깨는 유일한 방법이 기수 정렬법이다.
- '버킷'이라는 추가적인 메모리가 할당되어야 한다. 즉, 메모리가 엄청나게 여유롭다면 상관없겠지만 메모리가 생각보다 많이 소비되는 정렬법이다.
- 데이터 타입이 일정한 경우에만 가능하다. 기존에 정렬법들은 음수와 양수, 실수와 정수가 섞여 있더라도 비교를 하려면 할 수 있었지만, 기수 정렬의 경우 양의 정수는 양의 정수끼리만, 음의 정수는 음의 정수끼리만 정렬이 가능하다.
- 즉, 엄청나게 빠른 대신에 구현을 위한 조건이 굉장히 많이 붙기 때문에 그렇게 많이 사용되는 방법은 아니다.

2. 데이터 입, 출력 구현-SEC_03(정렬(Sort))

9) 정렬의 총정리

Sorting	장점	단점
버블 정렬	- 구현이 쉽다	- 시간이 오래 걸리고 비효율적이다.
선택 정렬	- 구현이 쉽다 - 비교하는 횟수에 비해 교환이 적게 일어난다.	- 시간이 오래 걸려서 비효율적이다.
퀵 정렬	- 실행시간 $O(N \log N)$ 으로 빠른 편이다	- Pivot에 의해서 성능의 차이가 심하다. - 최악의 경우 $O(N^2)$ 이 걸리게 된다.
힙 정렬	- 항상 $O(N \log N)$ 으로 빠른 편이다.	- 실제 시간이 다른 $O(N \log N)$ 이 정렬법들에 비해서 오래걸린다.
병합 정렬	- 항상 $O(N \log N)$ 으로 빠른 편이다.	- 추가적인 메모리 공간을 필요로 한다.
삽입 정렬	- 최선의 경우 $O(N)$ 으로 굉장히 빠른 편이다. - 성능이 좋아서 다른 정렬법에 일부로 사용됨.	- 최악의 경우 $O(N^2)$ 으로, 데이터의 상태에 따라서 성능 차이가 심하다.
셸 정렬	- 삽입정렬보다 더 빠른 정렬법이다.	- 설정하는 '간격'에 따라서 성능 차이가 심하다.
기수 정렬	- $O(N)$ 이라는 말도 안 되는 속도를 갖는다.	- 추가적인 메모리가 '많이' 필요하다. - 데이터 타입이 일정해야 한다. - 구현을 위한 조건이 많이 붙는다.

Sorting	최악의 경우(Worst)	일반(평균)적인 경우(Average)	최선의 경우(Best)
버블 정렬	$O(N^2)$	$O(N^2)$	$O(N^2)$
선택 정렬	$O(N^2)$	$O(N^2)$	$O(N^2)$
퀵 정렬	$O(N^2)$	$O(N \log N)$	$O(N \log N)$
힙 정렬	$O(N \log N)$	$O(N \log N)$	$O(N \log N)$
병합 정렬	$O(N \log N)$	$O(N \log N)$	$O(N \log N)$
삽입 정렬	$O(N^2)$	$O(N^2)$	$O(N)$
셸 정렬	$O(N^2)$	$O(N^{1.3, 1.5})$	$O(N)$
기수 정렬	$O(N)$	$O(N)$	$O(N)$

2. 데이터 입, 출력 구현-SEC_03(정렬(Sort)) 기출 및 예상 문제

기출 문제 및 예상 문제(정렬(Sort))

1. 다음 자료에 대하여 선택(Selection) 정렬을 이용하여
오름차순으로 정렬하고자 한다. 3회전 후의 결과로 옳은 것은?

37, 14, 17, 40, 35

- ① 14, 17, 37, 40, 35
- ② 14, 37, 17, 40, 35
- ③ 17, 14, 37, 35, 40
- ④ 14, 17, 35, 40, 37

설명 : 선택 정렬은 n개의 레코드 중에서 최소값을 찾아
첫 번째 레코드 위치에 놓고, 나머지(n-1)개 중에서 다시
최소값을 찾아서 두 번째 레코드 위치에 놓는 방식을 반복
하여 정렬하는 방법이다.

1회전 : 14,37,17,40,35 -> 14,37,17,40,35

2회전 : 14,17,37,40,35

3회전 : 14,17,35,40,37

4회전 : 14,17,35,37,40

2. 다음 초기 자료에 대하여 삽입 정렬(Insertion Sort)을
이용하여 오름차순 정렬할 경우 1회전 후의 결과는?

3. 다음 자료를 버블 정렬을 이용하여 오름차순으로 정렬할 경우
PASS 2의 결과는?

9, 6, 7, 3, 5

- ① 3, 5, 6, 7, 9
- ② 6, 7, 3, 5, 9
- ③ 3, 5, 9, 6, 7
- ④ 6, 3, 5, 7, 9

버블 정렬은 주어진 파일에서 인접한 2개의 레코드 키 값을 비교하여
그 크기에 따라 레코드 위치를 서로 교환하는 정렬 방식이다.

1회전 : 6,9,7,3,5 -> 6,7,9,3,5 -> 6,7,3,9,5 -> 6,7,3,5,9

2회전 : 6,7,3,5,9 -> 6,3,5,7,9

3회전 : 6,3,5,7,9 -> 3,6,5,7,9 -> 3,5,6,7,9

4회전 : 3,5,6,7,9

4. 퀵 정렬에 관한 설명으로 옳은 것은?

- ① 레코드의 키 값을 분석하여 같은 값끼리 그 순서에 맞는 버킷에
분배하였다가 버킷의 순서대로 레코드를 꺼내어 정렬한다.(기수정렬)
- ② 주어진 파일에서 인접한 두 개의 레코드 키 값을 비교하여 그 크기 에
따라 레코드 위치를 서로 교환한다.(버블정렬)

2. 데이터 입, 출력 구현-SEC_03(정렬(Sort)) 기출 및 예상 문제

기출 문제 및 예상 문제(정렬(Sort))

5. 힙 정렬(Heap Sort)에 대한 설명으로 틀린 것은?

- ① 정렬할 입력 레코드들로 힙을 구성하고 가장 큰 키 값을 갖는 루트 노드를 제거하는 과정을 반복하여 정렬하는 기법이다.
- ② 평균 수행 시간은 $O(N\log N)$ 이다.
- ③ 완전 이진 트리(Complete Binary Tree)로 입력자료의 레코드를 구성한다.
- ④ 최악의 수행 시간은 $O(2n^4)$ 이다.

설명 : 평균과 최악 수행 시간 복잡도 $O(N\log N)$ 으로 같은 정렬은 힙 정렬과 2-Way 합병정렬이 있다.

평균적인 시간 복잡도

$O(N^2)$: 버블 정렬, 선택 정렬, 삽입 정렬

$O(N\log N)$: 퀵 정렬, 힙 정렬, 병합 정렬

$O(N^{1.3, 1.5})$: 셸 정렬

$O(N)$: 기수 정렬

6. 정렬된 N개의 데이터를 처리하는 데 $O(N\log N)$ 의 시간이 소요되는 정렬 알고리즘은?

7. 레코드의 많은 자료 이동을 없애고 하나의 파일을 부분적으로 나누어 가면서 정렬하는 방법으로 키를 기준으로 작은 값은 왼쪽에, 큰 값은 오른쪽 서브 파일로 분해시키는 방식으로 정렬하는 것은?

- ① Selection Sort ② Bubble Sort
- ③ Insert Sort ④ Quick Sort

Selection Sort : n개의 레코드 중에서 최소값을 찾아 첫 번째 레코드 위치에 놓고, 나머지(n-1)개 중에서 다시 최소값을 찾아서 두 번째 레코드 위치에 놓는 방식을 반복하여 정렬하는 방법이다.

Bubble Sort : 주어진 파일에서 인접한 2개의 레코드 키 값을 비교하여 그 크기에 따라 레코드 위치를 서로 교환하는 정렬 방식이다.

Insert Sort : 가장 간단한 정렬 방식으로 이미 순서화된 파일에 새로운 하나의 레코드를 순서에 맞게 삽입시켜 정렬한다. 두 번째 키와 첫 번째 키를 비교해 순서대로 나열 (1회전)하고, 이어서 세 번째 키를 첫 번째, 두 번째 키와 비교해 순서대로 나열(2회전)하고, 계속해서 n번째 키를 앞의 n-1개의 키와 비교하며 알맞은 순서에 삽입하여 정렬하는 방식이다.

8. 입력 순서에 따라 배열된 5개의 데이터 (8, 3, 5, 2, 4)를 어떠한 정렬 방식에 의해 1단계 정렬시킨 결과가 2-8-5-3-4가 되었다면 사용된 정렬

2. 데이터 입, 출력 구현-SEC_04(검색 - 이분 검색/ 해싱)

1) 이분 검색

- 이분 검색(이진 검색, Binary Search)은 전체 파일을 두 개의 서브파일로 분리해 가면서 Key 레코드를 검색하는 방식이다.
- 이분 검색은 반드시 순서화된 파일이어야 검색할 수 있다.
- 찾고자 하는 Key 값을 파일의 중간 레코드 Key 값과 비교하면서 검색한다.
- 비교 횟수를 거듭할 때마다 검색 대상이 되는 데이터의 수가 절반으로 줄어들므로 탐색 효율이 좋고 탐색 시간이 적게 소요된다.
- 중간 레코드 번호 $M = \frac{(F+L)}{2}$ (단, F : 첫 번째 레코드 번호, L : 마지막 레코드 번호)

2. 데이터 입, 출력 구현-SEC_04(검색 - 이분 검색/ 해싱)

1) 이분 검색

예제) 1~100까지의 숫자 중 15를 찾는 데 걸리는 횟수는?

- ① 첫 번째 값(F)과 마지막 값(L)을 이용하여 중간 값 M을 구하여 찾으려는 값과 비교한다.

$$M = \frac{(1+100)}{2} = 50.5 \rightarrow 50(\text{정수만 취한다.})$$

- ② 50이 찾으려는 값과 같은지, 아니면 작는지, 아니면 큰지를 확인한다. 50은 찾으려는 값 보다 크다.
그러므로 찾으려는 값은 1~49 사이에 있다. → 1회 비교

- ③ 이제 첫 번째 값은 1이고 마지막 값은 49이다. 찾으려는 값이 50사이에 있지만 50은 아니므로 49가 마지막 값이 된다. 다시 중간 값을 구한다.

$$M = \frac{(1+49)}{2} = 25 \rightarrow 2\text{회 비교}$$

- ④ 25는 찾으려는 값 보다 크다. 그러므로 찾으려는 값은 1~24 사이에 있다. 다시 중간 값을 계산한다.

$$M = \frac{(1+24)}{2} = 12.5 \rightarrow 12 \rightarrow 3\text{회 비교}$$

- ⑤ 12는 찾으려는 값보다 작다. 그러므로 찾으려는 값은 13~24 사이에 있다.

$$M = \frac{(13+24)}{2} = 18.5 \rightarrow 18 \rightarrow 4\text{회 비교}$$

2. 데이터 입, 출력 구현-SEC_04(검색 - 이분 검색/ 해싱)

1) 이분 검색

⑥ 18은 찾으려는 값보다 크다. 그러므로 찾으려는 값은 13~17 사이에 있다.

$$M = \frac{(13+17)}{2} = 15 \rightarrow 15 \rightarrow 5\text{회 비교}$$

⑦ 15는 찾으려는 값과 같다.

※ 총 비교 횟수는 5회이다.

2) 해싱

; 해싱(Hashing)은 해시 테이블(Hash Table)이라는 기억공간을 할당하고, 해시 함수(Hash Function)를 이용하여 레코드 키에 대한 해시 테이블 내의 홈 주소(Home Address)를 계산한 후 주어진 레코드를 해당 기억장소에 저장하거나 검색 작업을 수행하는 방식이다.

; 해시 테이블(Hash Table)

- 레코드를 한 개 이상 보관할 있는 버킷들로 구성된 기억공간으로, 보조기억장치에 구성할 수도 있고 주 기억장치에 구성할 수도 있다.

2. 데이터 입, 출력 구현-SEC_04(검색-이분 검색/해싱)

2) 해싱

; 해시 테이블(Hash Table)

버킷 (Bucket)	① 하나의 주소를 갖는 파일의 한 구역을 의미한다. ② 버킷의 크기는 같은 주소에 포함될 수 있는 레코드 수를 의미한다.
슬롯(Slot)	한 개의 레코드를 저장할 수 있는 공간으로 n개의 슬롯이 모여 하나의 버킷을 형성한다.
Collision(충돌현상)	서로 다른 두 개 이상의 레코드가 같은 주소를 갖는 현상이다.
Synonym	충돌로 인해 같은 Home Address를 갖는 레코드들의 집합이다.
Overflow	계산된 Home Address의 Bucket 내에 저장할 기억공간이 없는 상태로, Bucket을 구성하는 Slot이 여러 개일 때 Collision은 발생해도 Overflow는 발생하지 않을 수 있다.

; 해싱 함수(Hashing Function)

제산법(Division)	레코드 키(K)를 해시표(Hash Table)의 크기보다 큰 수 중에서 가장 작은 소수(Prime, Q)로 나눈 나머지를 홈 주소로 삼는 방식, 즉 $h(k) = K \bmod Q$ 이다.
제곱법(Mid-Square)	레코드 키 값(K)을 제곱한 후 그 중간 부분의 값을 홈 주소로 삼는 방식이다.
폴딩법(Folding)	레코드 키 값(K)을 여러 부분으로 나눈 후 각 부분의 값을 더하거나 XOR(배타적 논리합)한 값을 홈 주소로 삼는 방식이다.
기수 변환법(Radix)	키 숫자의 진수를 다른 진수로 변환시켜 주소 크기를 초과한 높은 자릿수는 절단하고, 이를 다시 주소 범위에 맞게 조정하는 방법이다.
대수적 코딩법 (Algebraic Coding)	키 값을 이루고 있는 각 자리의 비트 수를 한 다항식의 계수로 간주하고, 이 다항식을 해시표의 크기에 의해 정의된 다항식으로 나누어 얻은 나머지 다항식의 계수를 홈 주소로 삼는 방식이다.
숫자 분석법(Digit Analysis, 계수 분석법)	키 값을 이루는 숫자의 분포를 분석하여 비교적 고른 자리를 필요한 만큼 택해서 홈 주소로 삼는 방식이다.
무작위법(Random)	난수(Random Number)를 발생시켜 나온 값을 홈 주소로 삼는 방식이다.

2. 데이터 입, 출력 구현- SEC_04(검색-이분 검색/해싱) 기출 및 예상 문제

기출 문제 및 예상 문제(검색-이분 검색/해싱)

1. 이진 검색 알고리즘에 대한 설명으로 틀린 것은?

- ① 탐색 효율이 좋고 탐색 시간이 적게 소요된다.
- ② 검색할 데이터가 정렬되어 있어야 한다.
- ③ 피보나치 수열에 따라 다음에 비교할 대상을 선정하여 검색한다.
- ④ 비교 횟수를 거듭할 때마다 검색 대상이 되는 데이터의 수가 절반으로 줄어든다.

설명 : 이진 검색은 전체 파일을 두 개의 서브파일로 분리해 가면서 Key 레코드를 검색하는 방식이다.

피보나치 수열이라는 것은 앞의 2개의 값으로 다음 값을 생성하는 수열이다. 1, 1, 2, 3, 5, 8, 13....

2. 해싱 함수(Hashing Function)의 종류가 아닌 것은?

- ① 제곱법(Mid-Square)
- ② 숫자 분석법(Digit Analysis)
- ③ 개방 주소법(Open Addressing)
- ④ 제산법(Division)

설명 : 해싱 함수는 7개가 존재한다.

3. 다음과 같이 레코드가 구성되어 있을 때, 이진 검색 방법으로 14를 찾을 경우 비교되는 횟수는?

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ① 2 ② 3
- ③ 4 ④ 5

$M = (1 + 15) / 2 \rightarrow 8$ - 1번 비교, 8이 14보다 작다. 9~15안에 있다.

$M = (9 + 15) / 2 \rightarrow 12$, - 2번 비교, 12는 14보다 작다. 13~15안에 있다.

$M = (13 + 15) / 2 \rightarrow 14$, - 3번 비교, 14는 찾는 값과 동일하다.

4. 해싱 함수 중 레코드 키를 여러 부분으로 나누고, 나눈 부분의 각 숫자를 더하거나 XOR 값을 홈 주소로 사용하는 방식은?

- ① 제산법 ② 폴딩법
- ③ 기수 변환법 ④ 숫자 분석법

제산법(Division) : 레코드 키(K)를 해시 테이블의 크기보다 큰 수 중에서 가장 작은 소수(Prime, Q)로 나눈 나머지를 홈 주소로 삼는 방식이다. $h(k) = K \bmod Q$ 이다

제곱법(Mid-Square) : 레코드 키(K)를 제곱한 후 그 중간 부분의 값을

2. 데이터 입, 출력 구현- SEC_04(검색-이분 검색/해싱) 기출 및 예상 문제

기출 문제 및 예상 문제(검색-이분 검색/해싱)

5. 알고리즘과 관련한 설명으로 틀린 것은?

- ① 주어진 작업을 수행하는 컴퓨터 명령어를 순서대로 나열한 것으로 볼 수 있다.
- ② 검색(Searching)은 정렬이 되지 않은 데이터 혹은 정렬이 된 데이터 중에서 키 값에 해당되는 데이터를 찾는 알고리즘이다.
- ③ 정렬(Sorting)은 흩어져 있는 데이터를 키 값을 이용하여 순서대로 열거하는 알고리즘이다.
- ④ 선형 검색은 검색을 수행하기 전에 반드시 데이터의 집합이 정렬되어 있어야 한다.

설명 : 선형 검색(Linear Search)은 정렬이 되어 있지 않은 파일에서 순차적으로 검색하는 방식으로, 찾고자 하는 키 값을 첫 번째 레코드 키 값부터 차례대로 비교하여 검색하는 방식을 의미한다.

6. 다음 중, 키 값을 이루는 숫자의 분포를 분석하여 비교적 고른 자리를 필요한 만큼 택해서 홈 주소로 삼는 방식의 해싱 함수는 무엇인가?

7. 다음 중, 해시 테이블에서 서로 다른 두 개 이상의 레코드가 같은 주소를 갖는 현상을 무엇이라 하는가?

- ① Bucket ② Collision
- ③ Slot ④ Synonym

8. 다음 중, 해시 테이블을 구성하는 자료구조로 알맞은 것은?

- ① 배열, 링크드 리스트 ② 배열, 벡터
- ③ 배열, 맵 ④ 배열, 큐

설명 : 해시 테이블은 배열과 링크드 리스트로 조합되어 구성되어진 저장 공간이다.

2. 데이터 입, 출력 구현-SEC_05(데이터베이스 개요)

1) 데이터저장소

; 데이터저장소는 소프트웨어 개발 과정에서 다루어야 할 데이터들을 논리적인 구조로 조직화하거나, 물리적인 공간에 구축한 것을 의미한다.

- 데이터저장소는 논리 데이터저장소와 물리 데이터저장소로 구분된다.
- 논리 데이터저장소는 데이터 및 데이터 간의 연관성, 제약조건을 식별하여 논리적인 구조로 조직화한 것을 의미한다.
- 물리 데이터저장소는 논리 데이터저장소에 저장된 데이터와 구조들을 소프트웨어가 운용될 환경의 물리적 특성을 고려하여 하드웨어적인 저장장치에 저장한 것을 의미한다.
- 논리 데이터저장소를 거쳐 물리 데이터저장소를 구축하는 과정은 데이터베이스를 구축하는 과정과 동일하다.

2. 데이터 입, 출력 구현-SEC_05(데이터베이스 개요)

2) 데이터베이스

; 데이터베이스는 특정 조직의 업무를 수행하는 데 필요한 상호 관련된 데이터들의 모임으로 다음과 같이 정의할 수 있다.

- **통합된 데이터(Integrated Data)** : 자료의 중복을 배제한 데이터의 모임이다.
- **저장된 데이터(Stored Data)** : 컴퓨터가 접근할 수 있는 저장 매체에 저장된 자료이다.
- **운영 데이터(Operational Data)** : 조직의 고유한 업무를 수행하는 데 존재 가치가 확실하고 없어서는 안 될 반드시 필요한 자료이다.
- **공용 데이터(Shared Data)** : 여러 응용 시스템들이 공동으로 소유하고 유지하는 자료이다.

데이터베이스의 정의는 여러 사람에 의해 공동으로 사용될 데이터를 중복을 배제하여 통합하고, 쉽게 접근하여 처리할 수 있도록 저장 장치에 저장하여 항상 사용할 수 있도록 운영하는 운영 데이터라고 생각하면 쉽다. 운영 데이터에서는 단순한 입·출력 자료나 작업 처리상 일시적으로 필요한 임시 자료는 운영 자료로 취급되지 않는다.

2. 데이터 입, 출력 구현-SEC_05(데이터베이스 개요)

3) DBMS(DataBase Management System : 데이터베이스 관리 시스템)

; DBMS란 사용자와 데이터베이스 사이에서 사용자의 요구에 따라 정보를 생성해 주고, 데이터베이스를 관리해 주는 소프트웨어이다.

- DBMS는 기존의 파일 시스템이 갖는 데이터의 종속성과 중복성의 문제를 해결하기 위해 제안된 시스템으로, 모든 응용 프로그램들이 데이터베이스를 공유할 수 있도록 관리해 준다.
- DBMS는 데이터베이스의 구성, 접근 방법, 유지관리에 대한 모든 책임을 진다.
- DBMS의 필수 기능에는 정의(Definition), 조작(Manipulation), 제어(Control) 기능이 있다.
 - **정의 기능** : 모든 응용 프로그램들이 요구하는 데이터 구조를 지원하기 위해 데이터베이스에 저장될 데이터의 형(Type)과 구조에 대한 정의, 이용 방식, 제약 조건 등을 명시하는 기능이다.
 - **조작 기능** : 데이터 검색, 갱신, 삽입, 삭제 등을 체계적으로 처리하기 위해 사용자와 데이터베이스 사이의 인터페이스 수단을 제공하는 기능이다.
 - **제어 기능**
 - ▶ 데이터베이스를 접근하는 갱신, 삽입, 삭제 작업이 정확하게 수행되어 데이터의 **무결성**이 유지되도록 제어해야 한다.
 - ▶ 정당한 사용자가 허가된 데이터만 접근할 수 있도록 보안(Security)을 유지하고 **권한(Authority)**을 **검사**할 수 있어야 한다.
 - ▶ 여러 사용자가 데이터베이스를 동시에 접근하여 데이터를 처리할 때 처리 결과가 항상 정확성을 유지하도록 **병행 제어(Concurrency Control)**를 할 수 있어야 한다.

2. 데이터 입, 출력 구현-SEC_05(데이터베이스 개요)

4) DBMS의 장, 단점

; DBMS의 장, 단점은 아래와 같다.

장점	단점
<ul style="list-style-type: none">• 데이터의 논리적, 물리적 독립성이 보장된다.• 데이터의 중복을 피할 수 있어 기억 공간이 절약 된다.• 저장된 자료를 공동으로 이용할 수 있다.• 데이터의 일관성을 유지할 수 있다.• 데이터의 무결성을 유지할 수 있다.• 보안을 유지할 수 있다.• 데이터를 표준화할 수 있다.• 데이터를 통합하여 관리할 수 있다.• 항상 최신의 데이터를 유지한다.• 데이터의 실시간 처리가 가능하다	<ul style="list-style-type: none">• 데이터베이스의 전문가가 부족하다.• 전산화 비용이 증가한다.• 대용량 디스크로의 집중적인 Access로 과부(Overhead)가 발생한다.• 파일의 예비(Backup)와 회복(Recovery)이 어렵다.• 시스템이 복잡하다.

; 데이터의 독립성은 종속성에 대비되는 말로 DBMS의 궁극적 목표이기도 하다. 데이터의 독립성에는 논리적 독립성과 물리적 독립성이 있다.

- **논리적 독립성** : 응용 프로그램과 데이터베이스를 독립시킴으로써, 데이터의 논리적 구조를 변경 시키더라도 응용 프로그램은 변경되지 않는다.
- **물리적 독립성** : 응용 프로그램과 보조기억장치 같은 물리적 장치를 독립시킴으로써, 데이터베이스 시스템의 성능 향상을 위해 새로운 디스크를 도입하더라도 응용 프로그램에는 영향을 주지 않고 데이터의 물리적 구조만을 변경한다.

2. 데이터 입, 출력 구현-SEC_05(데이터베이스 개요)

5) 스키마

; 스키마(Schema)는 데이터베이스의 구조와 제약 조건에 관한 전반적인 명세(Specification)를 기술 (Description)한 메타 데이터(Meta-Data)의 집합이다. 여기서 메타데이터는 '데이터에 관한 구조화된 데이터', '다른 데이터를 설명해 주는 데이터'로 이해하면 된다.

- 스키마는 데이터베이스를 구성하는 데이터 개체(Entity), 속성(Attribute), 관계(Relationship) 및 데이터 조작 시 데이터 값들이 갖는 제약 조건 등에 관해 전반적으로 정의한다.
- 스키마는 **사용자의 관점에 따라 외부 스키마, 개념 스키마, 내부 스키마**로 나뉘어진다.

외부 스키마	사용자나 응용 프로그래머가 각 개인의 입장에서 필요로 하는 데이터베이스의 논리적 구조를 정의한 것이다.
개념 스키마	<ul style="list-style-type: none">• 데이터베이스의 전체적인 논리적 구조로서, 모든 응용 프로그램이나 사용자들이 필요로 하는 데이터를 종합한 조직 전체의 데이터베이스로 하나만 존재한다.• 개체 간의 관계와 제약 조건을 나타내고, 데이터베이스의 접근 권한, 보안 및 무결성 규칙에 관한 명세를 정의한다.
내부 스키마	물리적 저장장치의 입장에서 본 데이터베이스 구조로서, 실제로 데이터베이스에 저장될 레코드의 형식을 정의하고 저장 데이터 항목의 표현 방법, 내부 레코드의 물리적 순서 등을 나타낸다.

2. 데이터 입, 출력 구현- SEC_05(데이터베이스 개요) 기출 및 예상 문제

기출 문제 및 예상 문제(데이터베이스 개요)

1. 데이터베이스의 정의 중 '데이터베이스는 어떤 조직의 고유 기능을 수행하기 위해 반드시 필요한 데이터를 의미 한다.'에 해당되는 것은?

- ① 통합된 데이터(Integrated Data)
- ② 저장 데이터(Stored Data)
- ③ 운영 데이터(Operational Data)
- ④ 공용 데이터(Shared Data)

설명

통합된 데이터 : 자료의 중복을 배제한 데이터의 모임

저장된 데이터 : 컴퓨터가 접근할 수 있는 저장 매체에 저장된 자료를 의미

운영 데이터 : 조직의 고유한 업무를 수행하는데 존재 가치가 확실하고 없어서는 안될 반드시 필요한 자료를 의미

공용 데이터 : 여러 응용 시스템들이 공동으로 소유하고 유지하는 자료를 의미

2. DBMS의 필수 기능 중 모든 응용 프로그램들이 요구하는 데이터 구조를 지원하기 위해 데이터베이스에 저장될

3. DBMS의 필수 기능 중 데이터베이스를 접근하여 데이터의 검색, 삽입, 삭제, 갱신 등의 연산 작업을 위한 사용자와 데이터베이스 사이의 인터페이스 수단을 제공하는 기능은?

- ① 정의 기능 ② 조작 기능
- ③ 제어 기능 ④ 절차 기능

데이터 검색, 갱신, 삽입, 삭제 용어 나오면 무조건 조작 기능이다.

4. 데이터베이스 관리 시스템(DBMS)의 주요 필수 기능과 거리가 먼 것은?

- ① 데이터베이스 구조를 정의할 수 있는 정의 기능
- ② 데이터 사용자의 통제 및 보안 기능
- ③ 데이터베이스 내용의 정확성과 안정성을 유지할 수 있는 제어 기능
- ④ 데이터 조작어로 데이터베이스를 조작할 수 있는 조작 기능

설명 : DBMS의 3개 기능으로 정의, 조작, 제어 암기!!!

2. 데이터 입, 출력 구현- SEC_05(데이터베이스 개요) 기출 및 예상 문제

기출 문제 및 예상 문제(데이터베이스 개요)

5. 데이터베이스 관리 시스템(DBMS)의 필수 기능 중 제어 기능에 대한 설명으로 거리가 먼 것은?

① 데이터베이스를 접근하는 갱신, 삽입, 삭제 작업이 정확하게 수행되어 데이터의 무결성이 유지되도록 제어해야 한다.

② 데이터의 논리적 구조와 물리적 구조 사이에 변환이 가능하도록, 두 구조 사이의 사상(Mapping)을 명시하여야 한다.

③ 정당한 사용자가 허가된 데이터만 접근할 수 있도록 보안(Security)을 유지하고 권한(Authority)을 검사할 수 있어야 한다.

④ 여러 사용자가 데이터베이스를 동시에 접근하여 데이터를 처리할 때 처리 결과가 항상 정확성을 유지하도록 병행 제어(Concurrency Control)를 할 수 있어야 한다.

② 데이터의 논리적 구조와 물리적 구조 사이에 변환이 가능하도록, 두 구조 사이의 사상(Mapping)을 명시하여야 한다는 것은 DBMS의 정의 해당된다.

6. 데이터베이스 구성의 장점이 아닌 것은?

① 데이터 중복 최소화

② 여러 사용자에게 의한 데이터 공유

③ 데이터 간의 종속성 유지

④ 데이터 내용의 일관성 유지

설명 : DBMS는 기존의 파일 시스템이 갖는 데이터의 종속성과 중복성의 문제를 해결하기 위해서 제안된 시스템으로, 모든 응용 프로그램들이 데이터베이스를 공유할 수 있도록 관리해 준다.

DBMS는 데이터베이스의 구성, 접근 방법, 유지관리에 대한 모든 책임을 진다. DBMS의 필수 기능에는 정의, 조작, 제어 기능이 있다.

7. 다음 설명에 해당하는 것은?

물리적 저장장치의 입장에서 본 데이터베이스 구조로서, 실제로 데이터베이스에 저장될 레코드의 형식을 정의하고 저장 데이터 항목의 표현 방법, 내부 레코드의 물리적 순서 등을 나타낸다.

① 외부 스키마

② 내부 스키마

③ 개념 스키마

④ 슈퍼 스키마

설명

2. 데이터 입, 출력 구현- SEC_05(데이터베이스 개요) 기출 및 예상 문제

기출 문제 및 예상 문제(데이터베이스 개요)

8. 다음에서 설명하는 스키마(Schema)는?

데이터베이스 전체를 정의한 것으로, 데이터 개체, 관계, 제약 조건, 접근 권한, 무결성 규칙 등을 명세한 것

- ① 개념 스키마 ② 내부 스키마
- ③ 외부 스키마 ④ 내용 스키마

데이터베이스 전체 정의, 데이터 개체, 관계, 제약 조건 등을 명세했다라고 하면 무조건 개념 스키마다.

9. 다음 중 데이터저장소에 대한 설명으로 옳지 않은 것은?

- ① 논리 데이터저장소는 데이터들을 논리적인 구조로 조직화한 것이다.
- ② 물리 데이터저장소는 논리 데이터저장소의 데이터와 구조를 하드웨어 저장장치에 저장한 것이다.
- ③ 물리 데이터저장소를 구축할 때는 소프트웨어가 운용될 환경의 물리적 특성을 고려해야 한다.
- ④ 데이터저장소의 구축 과정과 데이터베이스의 구축 과정은 상이하다.

10. DBMS(Database Management System)에 대한 설명으로 거리가 먼 것은?

- ① 사용자가 데이터베이스를 용이하게 관리할 수 있도록 지원하는 소프트웨어이다.
- ② 파일 시스템이 갖는 한계를 극복하기 위해 제안되었다.
- ③ 데이터베이스의 구성, 접근 방법, 유지관리에 대한 모든 책임을 진다.
- ④ 데이터베이스의 안정성을 위해 응용 프로그램이 데이터베이스를 공유하는 것을 제한한다.

DBMS는 데이터베이스를 이용하고자 하는 모든 응용 프로그램들이 데이터베이스를 공유할 수 있도록 관리해 주는 시스템이다.

11. 데이터베이스의 정의로 가장 적합한 것은?

- ① 공용 데이터(Shared Data), 통합 데이터(Integrated Data), 통신 데이터(Communication Data), 운영 데이터 (Operational Data)
- ② 공용 데이터 (Shared Data), 색인 데이터(Indexed Data), 통신 데이터(Communication Data), 운영 데이터(Operational Data)
- ③ 공용 데이터(Shared Data), 색인 데이터(Indexed Data), 저장 데이터(Stored Data), 운영 데이터(Operational Data)
- ④ 공용 데이터(Shared Data), 통합 데이터(Integrated Data), 저장

2. 데이터 입, 출력 구현- SEC_05(데이터베이스 개요) 기출 및 예상 문제

기출 문제 및 예상 문제(데이터베이스 개요)

12. DBMS의 필수 기능 중 정의 기능이 갖추어야 할 요건에 해당하는 것은?

- ① 데이터베이스를 접근하는 갱신, 삽입, 삭제 작업이 정확하게 수행되게 해야 한다.(조작 기능)
- ② 데이터와 데이터의 관계를 명확하게 명세할 수 있어야 하며, 원하는 데이터 연산은 무엇이든 명세할 수 있어야 한다.
- ③ 정당한 사용자가 허가된 데이터만 접근할 수 있도록 보안을 유지하여야 한다.(제어 기능)
- ④ 여러 사용자가 데이터베이스를 동시에 접근하여 처리할 때 데이터베이스와 처리 결과가 항상 정확성을 유지하도록 병행 제어를 할 수 있어야 한다.(제어 기능)

2. 데이터 입, 출력 구현-SEC_06(데이터 입, 출력)

1) 데이터 입·출력의 개요

; 데이터 입·출력은 소프트웨어의 기능 구현을 위해 데이터베이스에 데이터를 입력하거나 데이터베이스의 데이터를 출력하는 작업을 의미한다.

- 데이터 입·출력은 단순 입력과 출력뿐만 아니라 데이터를 조작하는 모든 행위를 의미하며, 이와 같은 작업을 위해 SQL(Structured Query Language)을 사용한다.
- 데이터 입·출력을 소프트웨어에 구현하기 위해 개발 코드 내에 SQL 코드를 삽입하거나, 객체와 데이터를 연결하는 것을 데이터 접속(Data Mapping)이라고 한다.
- SQL을 통한 데이터베이스의 조작을 수행할 때 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산들을 트랜잭션(Transaction)이라고 한다.

2. 데이터 입, 출력 구현-SEC_06(데이터 입, 출력)

2) SQL(Structured Query Language)

; SQL은 1974년 IBM 연구소에서 개발한 SEQUEL에서 유래한다. 국제표준 데이터베이스 언어로, 많은 회사에서 관계형 데이터베이스(RDB)를 지원하는 언어로 채택하고 있다.

- 관계형 데이터베이스(RDB, Relational DataBase)는 2차원적인 표(Table)를 이용해서 데이터 상호 관계를 정의하는 데이터베이스이다.
- 관계대수와 관계해석을 기초로 한 혼합 데이터 언어이다.
- 질의어지만 질의 기능만 있는 것이 아니라 데이터 구조의 정의, 데이터 조작, 데이터 제어 기능을 모두 갖추고 있다.
- SQL은 데이터 정의어(DDL), 데이터 조작어(DML), 데이터 제어어(DCL)로 구분된다.
 - **데이터 정의어(DDL: Data Define Language)** : SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 정의하거나 변경 또는 삭제할 때 사용하는 언어이다.
 - **데이터 조작어(DML: Data Manipulation Language)** : 데이터베이스 사용자가 응용프로그램이나 질의어를 통하여 저장된 데이터를 실질적으로 처리하는 데 사용되는 언어이다.
 - **데이터 제어어(DCL: Data Control Language)** : 데이터의 보안, 무결성, 회복, 병행 수행 제어 등을 정의하는 데 사용되는 언어이다.

관계대수는 관계형 데이터베이스에서 원하는 정보와 그 정보를 검색하기 위해서 어떻게 유도하는가를 기술하는 절차적인 언어이고 관계해석은 관계 데이터의 연산을 표현하는 방법으로 원하는 정보를 정의할 때는 계산 수식을 사용한다

2. 데이터 입, 출력 구현-SEC_06(데이터 입, 출력)

3) 데이터 접속(Data Mapping)

; 데이터 접속은 소프트웨어의 기능 구현을 위해 프로그래밍 코드와 데이터베이스의 데이터를 연결 (Mapping)하는 것을 말하며, 관련 기술로 SQL Mapping과 ORM이 있다.

- SQL Mapping : 프로그래밍 코드 내에 SQL을 직접 입력하여 DBMS의 데이터에 접속하는 기술로, 관련 프레임워크에는 JDBC, ODBC, MyBatis 등이 있다.
- ORM(Object-Relational Mapping) : 객체지향 프로그래밍의 객체(Object)와 관계형(Relational) 데이터베이스의 데이터를 연결(Mapping)하는 기술로, 관련 프레임 워크에는 JPA, Hibernate, Django 등이 있다.

JDBC(Java Database Connectivity) : Java 기반 애플리케이션의 데이터를 데이터베이스에 저장 및 업데이트하거나, 데이터베이스에 저장된 데이터를 Java에서 사용할 수 있도록 하는 자바 API이다.

JDBC는 Java 애플리케이션에서 데이터베이스에 접근하기 위해 JDBC API를 사용하여 데이터베이스에 연동할 수 있으며, 데이터베이스에서 자료를 쿼리(Query)하거나 업데이트하는 방법을 제공한다.

ODBC(open database connectivity) : 어떤 응용프로그램을 사용하는지에 관계없이, 데이터베이스를 자유롭게 사용하기 위하여 MS사에서 만든 응용프로그램의 표준방법을 말한다. DBMS에 관계없이 어떤 응용 프로그램에서나 모두 접근하여 사용할 수 있도록 하기 위하여, 응용 프로그램과 DBMS 중간에 데이터베이스 처리 프로그램을 두어 이를 가능하게 한다.

MyBatis : 객체 지향 언어인 자바의 관계형 데이터베이스 프로그래밍을 좀 더 쉽게 할 수 있게 도와 주는 개발 프레임 워크로서 JDBC를 통해 데이터베이스에 액세스하는 작업을 캡슐화하고 일반 SQL 쿼리, 저장 프로시저 및 고급 매핑을 지원하며 모든 JDBC 코드 및 매개 변수의 중복작업을 제거한다. Mybatis에서는 프로그램에 있는 SQL쿼리들을 한 구성파일에 구성하여 프로그램 코드와 SQL을 분리할 수 있는 장점을 가지고 있다.

JPA(Java Persistence API) : 현재 자바 진영의 ORM 기술 표준으로, 인터페이스의 모음이다.

Hibernate : JPA 인터페이스를 구현한 대표적인 오픈 소스

Django : 파이썬으로 만들어진 무료 오픈 소스 웹 애플리케이션 프레임워크(web application framework)이다.

2. 데이터 입, 출력 구현-SEC_06(데이터 입, 출력)

4) 트랜잭션(Transaction)

; 트랜잭션은 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산들을 의미한다.

● 트랜잭션을 제어하기 위해서 사용하는 명령어를 TCL(Transaction Control Language)이라고 하며, TCL의 종류에는 COMMIT, ROLLBACK, SAVEPOINT가 있다.

- **COMMIT** : 트랜잭션 처리가 정상적으로 종료되어 트랜잭션이 수행한 변경 내용을 데이터베이스에 반영하는 명령어
- **ROLLBACK** : 하나의 트랜잭션 처리가 비정상적으로 종료되어 데이터베이스의 일관성이 깨졌을 때 트랜잭션이 행한 모든 변경 작업을 취소하고 이전 상태로 되돌리는 연산
- **SAVEPOINT**(=CHECKPOINT) : 트랜잭션 내에 ROLLBACK 할 위치인 저장점을 지정하는 명령어, SAVEPOINT로 저장점을 지정해 두면 해당 위치까지만 ROLLBACK을 수행하여 저장점 이전에 수행한 작업에는 영향을 주지 않고 저장점 이후에 수행한 작업만을 취소하고 이전 상태로 되돌릴 수 있다.

2. 데이터 입, 출력 구현-SEC_06(데이터 입, 출력) 기출 및 예상 문제

기출 문제 및 예상 문제(데이터 입, 출력)

1. 트랜잭션(Transaction)에 대한 설명으로 옳지 않은 것은?

- ① 트랜잭션은 작업의 논리적 단위이다.
- ② 트랜잭션을 제어하기 위한 명령어를 TCL이라고 한다.
- ③ 하나의 트랜잭션은 Commit되거나 Rollback되어야 한다.
- ④ Savepoint는 트랜잭션당 한 번만 지정할 수 있다.

트랜잭션은 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산을 의미한다.

트랜잭션을 제어하기 위해서 사용하는 명령어가 TCL이다.

COMMIT : 트랜잭션 처리가 정상적으로 종료되어 트랜잭션이 수행한 변경 내용을 물리적인 데이터베이스에 반영하는 명령어

ROLLBACK : 하나의 트랜잭션 처리가 비정상적으로 종료되어 데이터베이스의 일관성이 깨졌을 때 트랜잭션이 행한 모든 변경 작업을 취소하고 이전 상태로 되돌리는 연산을 하는 명령어

SAVEPOINT(=CHECKPOINT) : 트랜잭션 내에 ROLLBACK 할

2. 데이터 접속(Data Mapping)에 대한 설명 중 보기에 해당하는 기술은?

- 객체지향 프로그래밍의 객체(Object)와 관계형(Relational) 데이터베이스의 데이터를 연결(Mapping)하는 기술이다.
- 부수적인 코드가 생략되고 SQL 코드를 직접 입력할 필요가 없어 간단하고 직관적인 코드로 데이터를 조작할 수 있다.
- 관련 프레임워크에는 JPA, Hibernate, Django 등이 있다.

① ORM

② SQL Mapping

③ JDBC

④ ODBC

3. 데이터 입·출력에 대한 설명으로 거리가 먼 것은?

① 응용 프로그램이 데이터베이스로부터 데이터를 입·출력하는 작업을 의미한다.

② 데이터를 조작하는 행위를 제외한 소프트웨어와 데이터베이스 간의 데이터 전송만을 의미한다.

③ 데이터 입·출력에 필요한 일련의 연산들이 포함된 하나의 작업 단위를 트랜잭션이라고 한다.

④ 데이터 입·출력을 위해 응용 프로그램의 객체와 데이터베이스의 데이터를 연결하는 것을 데이터 매핑이라고 한다.

데이터 입·출력은 단순 입력과 출력뿐만 아니라 데이터를 조작하는 모든

2. 데이터 입, 출력 구현-SEC_06(데이터 입, 출력) 기출 및 예상 문제

기출 문제 및 예상 문제(데이터 입, 출력)

4. 트랜잭션을 제어하기 위해 사용하는 명령어인 TCL의 종류에 속하지 않는 것은?

- ① COMMIT ② RETURN
- ③ ROLLBACK ④ SAVEPOINT

5. 다음 SQL(Structured Query Language)에 대한 설명으로 옳지 않은 것은?

- ① SQL은 데이터 구조의 정의, 조작, 제어 기능을 갖춘 혼합 데이터 언어로, DDL, DML, DCL로 구성되어 있다.
 - ② DDL은 데이터베이스에 문제가 발생했을 때 복원하는 작업을 수행한다.
 - ③ DML은 튜플에 대한 조회, 삽입, 삭제 등의 작업을 수행한다.
 - ④ DCL은 보안, 무결성, 병행 제어 등의 작업을 수행한다.
- SQL은 데이터 정의어(DDL), 데이터 조작어(DML), 데이터 제어어(DCL)로 구분된다.
- 튜플(tuple)은 레코드나 행(row)과 같은 개념이다.

6. 다음 중, 프로그래밍 코드 내에 SQL을 직접 입력하여 DBMS의 데이터에 접속하는 기술로, 관련 프레임워크에 해당하지 않는 것은?

- ① ORM ② JDBC
- ③ ODBC ④ MyBatis

SQL Mapping : 프로그래밍 코드 내에 SQL을 직접 입력하여 DBMS의 데이터에 접속하는 기술로, 관련 프레임워크에는 JDBC, ODBC, MyBatis가 있다.

2. 데이터 입, 출력 구현-SEC_07(절차형 SQL)

1) 절차형 SQL의 개요

; 절차형 SQL은 C, JAVA 등의 프로그래밍 언어와 같이 연속적인 실행이나 분기, 반복 등의 제어가 가능한 SQL을 의미한다. DBMS 벤더 별로 PL(Procedural Language)/SQL(Oracle), SQL/PL(DB2), T-SQL(SQL Server) 등의 절차형 SQL을 제공하고 있다.

- 절차형 SQL은 일반적인 프로그래밍 언어에 비해 효율은 떨어지지만 단일 SQL 문장으로 처리하기 어려운 연속적인 작업들을 처리하는데 적합하다.
- 절차형 SQL을 활용하여 다양한 기능을 수행하는 저장 모듈을 생성할 수 있다.
- 절차형 SQL은 DBMS 엔진에서 직접 실행되기 때문에 입·출력 패킷이 적은 편이다.
- BEGIN END 형식으로 작성되는 블록(Block)구조로 되어 있기 때문에 기능별 모듈화가 가능하다.
- 절차형 SQL의 종류에는 프로시저, 트리거, 사용자 정의 함수가 있다.
 - **프로시저(Procedure)** : 특정 기능을 수행하는 일종의 트랜잭션 언어로, 호출을 통해 실행되어 미리 저장해 놓은 SQL 작업을 수행한다.
 - **트리거(Triiger)** : 데이터베이스 시스템에서 데이터의 입력, 갱신, 삭제 등의 이벤트(Event)가 발생할 때마다 관련 작업이 자동으로 수행된다.
 - **사용자 정의 함수** : 프로시저와 유사하게 SQL을 사용하여 일련의 작업을 연속적으로 처리하며, 종료 시 예약어 Return을 사용하여 처리 결과를 단일 값으로 반환한다.

2. 데이터 입, 출력 구현-SEC_07(절차형 SQL)

2) 절차형 SQL의 테스트와 디버깅

; 절차형 SQL은 디버깅을 통해 기능의 적합성 여부를 검증하고, 실행을 통해 결과를 확인하는 테스트 과정을 수행한다.

- 절차형 SQL은 테스트 전에 생성을 통해 구문 오류(Syntax Error)나 참조 오류의 존재 여부를 확인한다.
- 많은 코드로 구성된 절차형 SQL의 특성상 오류 및 경고 메시지가 상세히 출력되지 않으므로 SHOW 명령어를 통해 내용을 확인하고 문제를 수정한다.
- 정상적으로 생성된 절차형 SQL은 디버깅을 통해 로직을 검증하고, 결과를 통해 최종적으로 확인한다.
- 절차형 SQL의 디버깅은 실제로 데이터베이스에 변화를 줄 수 있는 삽입 및 변경 관련 SQL문을 주석으로 처리하고, 출력문을 이용하여 화면에 출력하여 확인한다.

SHOW는 데이터베이스 목록이나, 테이블 목록 등 다양한 정보를 보기 원할 때 사용하는 명령어이다.

테스트와 디버깅의 목적 : 테스트(Test)를 통해 오류를 발견한 후 디버깅(Debugging)을 통해 오류가 발생한 소스 코드를 추적 하며 수정한다.

구문 오류(Syntax Error) : 구문 오류란 잘못된 문법으로 작성된 SQL문을 실행하면 출력되는 오류를 말한다.

2. 데이터 입, 출력 구현-SEC_07(절차형 SQL)

3) 쿼리 성능 최적화

; 쿼리 성능 최적화는 데이터 입·출력 애플리케이션의 성능 향상을 위해 SQL 코드를 최적화하는 것이다.

- 쿼리 성능을 최적화하기 전에 성능 측정 도구인 APM을 사용하여 최적화 할 쿼리를 선정해야 한다.
- 최적화 할 쿼리에 대해 옵티마이저가 수립한 실행 계획을 검토하고 SQL 코드와 인덱스를 재구성한다.

SQL을 통해 데이터베이스에 정보를 요청하는 것을 질의 또는 쿼리라고 한다.

APM(Application Performance Management/Monitoring) : APM은 애플리케이션의 성능 관리를 위해 접속자, 자원 현황, 트랜잭션 수행 내역, 장애 진단 등 다양한 모니터링 기능을 제공하는 도구이다.

옵티마이저(Optimizer) : 옵티마이저는 DBMS에 내장되어 작성된 SQL이 효율적으로 수행되도록 최적의 경로를 찾아 주는 모듈이다.

2. 데이터 입, 출력 구현- SEC_07(절차형 SQL) 기출 및 예상 문제

기출 문제 및 예상 문제(절차형 SQL)

1. 절차형 SQL에서 테스트와 디버깅의 목적으로 옳은 것은?

- ① 테스트는 오류를 찾는 작업이고 디버깅은 오류를 수정하는 작업이다.
- ② 테스트는 오류를 수정하는 작업이고 디버깅은 오류를 찾는 작업이다.
- ③ 둘 다 소프트웨어의 오류를 찾는 작업으로 오류 수정은 하지 않는다.
- ④ 둘 다 소프트웨어 오류의 발견, 수정과 무관하다.

테스트와 디버깅의 목적 : 테스트를 통해서 오류를 발견한 후 디버깅(Debugging)을 통해 오류가 발생한 소스 코드를 추적하며 수정한다.

2. 절차형 SQL에 대한 설명으로 옳지 않은 것은?

- ① 절차형 SQL의 종류에는 프로시저, 트리거, 사용자 정의 함수가 있다.
- ② 프로시저는 특정 기능을 수행하는 트랜잭션 언어로, 처리 결과를 단일값으로 반환한다.
- ③ 트리거는 데이터베이스에 이벤트가 발생할 때 수행되는

3. 절차형 SQL의 생성부터 최적화까지의 과정에 대한 설명으로 거리가 먼 것은?

- ① 절차형 SQL을 생성할 때 오류가 발생했다면 SHOW 명령을 통해 오류 내용을 확인한다.
- ② 절차형 SQL을 실행하기 전에 디버깅을 통해 로직을 검증한다.
- ③ 디버깅 시 데이터베이스의 데이터들이 변경되지 않도록 관련 코드들을 주석으로 처리한다.
- ④ 절차형 SQL의 성능이 느리다면 사용된 SQL 코드 중 가장 긴 SQL 코드의 최적화를 수행한다.

SQL코드가 길다고 무조건 비효율적인 쿼리는 아니다. 최적화는 성능 측정 도구인 APM을 사용하여 각 쿼리의 성능을 확인한 후 성능이 떨어지는 쿼리를 대상으로 최적화를 수행한다.

APM(Application Performance Management/Monitoring) : APM은 애플리케이션의 성능 관리를 위해 접속자, 접속 현황, 트랜잭션 수행 내역, 장애 진단 등 다양한 모니터링 기능을 제공하는 도구이다.

4. 절차형 SQL의 테스트에 대한 설명으로 잘못된 것은?

- ① 디버깅과 실행을 통한 결과 검증으로 테스트를 수행한다.
- ② 디버깅 시 데이터베이스에 변화를 주는 코드들은 모두 삭제한 후 변경

2. 데이터 입, 출력 구현- SEC_07(절차형 SQL) 기출 및 예상 문제

기출 문제 및 예상 문제(절차형 SQL)

5. 다음 중, DBMS에 내장되어 작성된 SQL이 효율적으로 수행되도록 최적의 경로를 찾아 주는 모듈을 무엇이라고 하는가?

- ① Query ② Function
- ③ Optimizer ④ APM

SQL을 통해 데이터베이스에 정보를 요청하는 것을 질의 또는 쿼리라고 한다.

옵티마이저 : DBMS에 내장되어 작성된 SQL이 효율적으로 수행되도록 최적의 경로를 찾아 주는 모듈

6. 절차형 SQL의 대한 내용으로 틀린 것은?

- ① 절차형 SQL을 활용하여 다양한 기능을 수행하는 저장 모듈을 생성할 수 있다.
- ② 절차형 SQL은 DBMS 엔진에서 직접 실행되기 때문에 입·출력 패킷이 많은 편이다.
- ③ BEGIN~END 형식으로 작성되는 블록(Block)구조로 되어 있기 때문에 기능별 모듈화가 가능하다.
- ④ 절차형 SQL의 종류에는 프로시저, 트리거, 사용자 정의



감사합니다.