

[25~26] 다음 글을 읽고 물음에 답하시오. -2011.11

소프트웨어 개발에서 자료 관리를 위한 구조로는 ‘배열’과 ‘연결 리스트’가 흔히 사용된다. 이 구조를 가진 저장소가 실제 컴퓨터 메모리에 구현된 위치를 ‘포인터’라고 한다.

㉠ 배열은 물리적으로 연속된 저장소들을 사용한다. 배열에서는 흔히 <그림 1>과 같이 자료의 논리적 순서와 실제 저장 순서가 일치하도록 자료가 저장된다. 이때 원하는 자료의 논리적인 순서만 알면 해당 포인터 값을 계산할 수 있으므로, 바로 접근하여 읽기와 쓰기를 할 수 있다. 그런데 <그림 1>에서 자료 ‘지리’를 삭제하려면 ‘한라’를 한 칸 당겨야 하고, 가나다순에 따라 ‘소백’을 삽입하려면 ‘지리’부터 한 칸씩 밀어야 한다. 따라서 삽입하거나 삭제하는 자료의 순번이 빠를수록 나머지 자료의 재정렬 시간이 늘어난다.

포인터:	저장소	포인터:	저장소
0000:	산 이름	0000:	산 이름 다음 포인터
1000:	백두	1000:	백두 1008
1001:	설악	1002:	㉠ ㉡
1002:	지리	1004:	지리 1006
1003:	한라	1006:	한라 ----
1004:		1008:	설악 ㉢1004
:		:	

<그림 1> 배열

<그림 2> 연결 리스트

㉡ 연결 리스트는 저장될 자료와 다음에 올 자료의 포인터인 ‘다음 포인터’를 한 저장소에 함께 저장한다. 이 구조에서는 <그림 2>와 같이 ‘다음 포인터’의 정보를 담은 공간이 더 필요하지만, 이 정보에 의해 물리적 저장 위치에 상관없이 자료의 논리적 순서를 유지할 수 있다. 또한 자료의 삽입과 삭제는 ‘다음 포인터’의 내용 변경으로 가능하므로 상대적으로 간단하다. 예를 들어 <그림 2>에서 ‘소백’을 삽입하려면 빈 저장소의 ㉠에 ‘소백’을 쓰고 ㉡와 ㉢에 논리적 순서에 따라 다음에 올 포인터 값인 ‘1004’와 ‘1002’를 각각 써 주면 된다. 하지만 특정 자료를 읽으려면 접근을 시작하는 포인터부터 그 자료까지 저장소들을 차례로 읽어야 하므로 자료의 논리적 순서에 따라 접근 시간에 차이가 있다.

한편 ‘다음 포인터’뿐만 아니라 논리순으로 앞에 연결된 저장소의 포인터를 하나 더 저장하는 ㉢ ‘이중 연결 리스트’도 있다. 이 구조에서는 현재 포인터에서부터 앞뒤 어느 방향으로든 연결된 자료에 접근할 수 있어 연결 리스트보다 자료 접근이 용이하다.

25. 윗글을 통해 알 수 있는 사실로 옳지 않은 것은?

- ① 저장된 자료에 접근할 때는 포인터를 이용한다.
- ② 자료 접근 과정은 사용하는 자료 관리 구조에 따라 달라진다.
- ③ ‘배열’에서는 자료의 논리적 순서에 따라 자료 접근 시간이 달라진다.
- ④ ‘연결 리스트’는 저장되는 전체 자료의 개수가 자주 변할 때 편리하다.
- ⑤ ‘이중 연결 리스트’의 한 저장소에는 세 가지 다른 정보가 저장된다.

26. ㉠~㉢에 대해 <보기>의 실험을 한 후 얻은 결과로 옳은 것은? [3점]

<보 기>

동일 수의 자료를 논리순이 유지되도록 메모리에 저장한 다음 읽기, 삽입, 삭제를 동일 횟수만큼 차례로 실행하였다.

\* 단, 충분히 많은 양의 자료로 충분한 횟수만큼 실험을 하되, 자료를 무작위로 선택하고 자료의 논리순이 유지되도록 함.

- ① ㉠은 ㉡에 비해 삭제 실험에 걸리는 총시간이 길었다.
- ② ㉠은 ㉢에 비해 저장 실험의 메모리 사용량이 많았다.
- ③ ㉡은 ㉠에 비해 삽입 실험에 걸리는 총시간이 길었다.
- ④ ㉡은 ㉢에 비해 저장 실험의 메모리 사용량이 많았다.
- ⑤ ㉢은 ㉡에 비해 읽기 실험에 걸리는 총시간이 길었다.