적용학습 기술 1. 스택 알고리즘의 이해와 적용 사용설명서 완벽암기

***지문 확인 문제**

※ 다음 글을 읽고, 빈 칸에 알맞은 말을 쓰시오.

컴퓨터에서는 데이터를 처리하기 위해 여러 가지 데이터 구조를 활용한다. 그 대표적인 것으로 '스택(stack)'을 들 수 있다. 스택은 한쪽 끝에서만 데이터가 들어가거나 나올 수 있는 특성을 가지는데, 스택에서 데이터가 가장 먼저 나올 수 있는 위치를 톱(top)이라 한다. 스택의 구조는 위쪽만 뚫려 있는 통 안에 책을 차곡차곡 쌓는다고 생각하면 쉽게 이해할 수 있다. 만약 n개의 책을 통에 쌓았다면 가장 먼저 꺼낼 수 있는 책은 아래에서부터 n번째에 있는 것이고, 그 위에 책을 또 올리게 되면 그 책은 아래에서부터 n+1번째에 있게 된다. 스택 구조는 이와 같은 통에, 톱은 가장 위에 있는 책의 위치에 비유될 수 있다. 그리고 스택 알고리즘은 데이터가 들어가 보관되고 나오는 규칙으로, 통에 책을 넣어 쌓고 책을 꺼내는 행위에 비유될 수 있다.

컴퓨터 내의 데이터 처리에서 스택 알고리즘은 수식 변환의 과정에 활용된다. 수식은 '+'(덧셈), '*'(곱 셈), '−'(뺄셈), '/'(나눗셈) 등과 같은 연산자와, 'x, y, z, A, B, C' 등과 같은 변수나 '1, 2, 3' 등과 같은 상수가 포함되는 피연산자로 구성된다. 그리고 수식은 크게 세 가지 표기 식으로 표현할 수 있는데, 연산자가 피연산자 가운데 있는 중위 표기 식, 연산자가 피연산자 앞에 있는 전위 표기 식, 연산자가 피연산자 뒤에 있는 후위 표기 식이 그것이다. 예를 들어 'x 더하기 y 나누기 z'는 중위 표기 식으로는 'x +y/z'로, 전위 표기 식으로는 '+x/yz'로, 후위 표기 식으로는 'xyz/ +'로 표현된다. 우리가 일상생활에서 사용하는 것은 중위 표기 식이지만, 컴퓨터 내부에서는 괄호를 사용하지 않고도 계산해야 할 순서를 알 수 있는 후위 표기 식을 활용하기 때문에 중위 표기 식으로 입력한 수식은 스택 알고리즘을 통해 후위 표기 식으로 변환되어 계산된다.

중위 표기 식을 후위 표기 식으로 변환하는 방법은 간단하다. 기본적인 전제는 피연산자들의 순서는 변하지 않는다는 것이다. 즉 중위 표기 식을 연산자 기준으로 괄호로 묶은 다음, 각 연산자를 묶고 있는 괄호의 오른쪽 괄호 밖으로 연산자를 이동한 후 괄호를 모두 제거하면 된다. 예를 들어 중위 표기 식 'A + B*C'를 후위 표기 식으로 바꾸기 위해서는 '(A +(B*C))'로 괄호를 묶은 뒤, '(A(BC)*) +'와 같이 괄호의 오른쪽에 연산자를 옮긴 후 'ABC* +'와 같이 괄호를 제거하면 된다. 중위 표기 식을 전위 표기 식으로 변환하는 과정 역시 동일하다. 다만 연산자를 오른쪽에 옮기는 것이 아니라 왼쪽으로 옮기는 것만 차이가 있다.

컴퓨터에서는 중위 표기 식을 후위 표기 식으로 변환하기 위해 앞에서 언급한 스택 알고리즘을 활용한다. 후위 표기 식에서 피연산자들의 순서는 변하지 않으므로, 선후 순서가 바뀌어야 되는 연산자들만 다른 위치로 이동하면 된다. 스택 알고리즘에서 피연산자는 순서가 되면 후위 표기로 바로 나오게 된다. 그러나 연산자의 경우 피연산자와 달리 뒤에 스택으로 들어갈 연산자의 존재 여부나 우선순위에 따라 스택에 보관되거나 후위 표기로 나오는 것이 결정된다. 이때 연산자의 우선순위는 괄호 안의 연산을 우선하되, 덧셈과 뺄셈보다 곱셈과 나눗셈을 우선하는 것이 원칙이다. 예를 들어 컴퓨터에서 스택 알고리즘을 활용해 중위 표기 식 'A+B*C'를 후위 표기 식으로 바꾸는 경우, 들어가는 데이터는 'A', '+', 'B', '*', 'C'로 모두 5개이다. 들어가는 5개의 데이터 중 피연산자들은 스택에 보관되지 않고 후위 표기에 순차적으로 나오고, 연산자들은 스택에 순차적으로 보관된다. 다만 연산자들이 스택에 보관되거나 후위 표기로 나올 때에는 현재의 스택 톱에 있는 연산자와 우선순위가 비교되는 과정을 먼저 거친다. 만일 스택 톱에 있는 연산자의 우선순위가 보관되려는 연산자의 우선순위보다 높거나 같으면 스택 톱에 있는 연산자가 먼저 스택에서 나오게 된다. 그런 뒤에는 다시 스택 톱에 오게 되는 다른 연산자와 우선순위가 비교된다. 중위 표기를 다 읽고 나면 연산자가 존재하지 않아 스택에 남아 있는 모든 연산자가 톱에서부터 하나씩 나오게 된다.

1. 데이터를 처리하기 위해 활용하는 데이터 구조 중 (　　)은 한쪽 끝에서만 데이터가 들어가거나 나올 수 있으며, 데이터가 가장 먼저 나올 수 있는 위치를 (　　)이라고 한다.

2. 스택 알고리즘은 데이터가 (　　) 규칙으로, (　　) 행위에 비유될 수 있다.

3. 스택 알고리즘은 ( )의 과정에 활용되는데, 수식은 ( )와 ( )로 구성된다.

4. 'X+Y/Z'는 ( )개의 연산자와 ( )개의 피연산자로 구성된 ( ) 표기 식이다.

5. 우리가 일상에서 사용하는 ( ) 표기 식은 컴퓨터 내부에서 ( ) 표기 식으로 변환되어 계산되는데, 그 이유는 ( )를 사용하지 않고도 ( )를 알 수 있기 때문이다.

6. 중위 표기 식을 후위 표기 식으로 변환할 때에는 ( )를 기준으로 ( )로 묶은 뒤, 연산자를 묶고 있는 괄호의 (오른쪽) 밖으로 연산자를 이동 후 괄호를 모두 제거하면 된다. 만약 전위 표기 식으로 변환하려면 괄호의 (왼쪽) 밖으로 연산자를 이동하면 된다.

7. 컴퓨터에서 중위 표기 식을 후위 표기 식으로 변환하기 위해서 ( )을 활용하는데, 이때 후위 표기 식에서 ( )의 순서는 변하지 않고, ( )만 선후 순서가 바뀌어 이동된다.

8. 연산자는 뒤에 스택으로 들어갈 ( )의 존재여부나 ( )에 따라 ( )에 보관될 수도 있고, ( )로 나올 수도 있다.

9. 연산자의 우선순위는 ( )의 연산을 우선하되, ( )보다 ( )이 우선하는 것이 원칙이며, 우선순위가 같을 경우에는 ( ) 에 있는 연산자가 먼저 ( )에서 나오게 된다.

10. 'A+B*C'를 스택 알고리즘을 통해 후위 표기 식으로 변환할 때 피연산자인 (①)의 순서는 변하지 않고, 연산자인 (②)의 순서는 변하는데, 이때 우선순위에 따라 연산자 중 (③)가 먼저 스택에서 나오게 된다.

11. 후위 표기 식으로 변환할 때 스택 알고리즘이 ( ) 표기를 다 읽고 나면 더이상 ( )가 존재하지 않으므로 ( )에 남아있던 연산자들이 ( )에서부터 하나씩 나오게 된다.

12.

< 스택 >

- 컴퓨터에서 사용되는 ( ) 중 하나.
- ( ) 끝에서만 데이터가 들어가고 나오는 구조
- 제일 나중에 들어간 데이터가 제일 ( ) 나옴.

13.

< 수식의 세 가지 표기 식 >

- 중위 표기 식 : 연산자가 피연산자 ( ) 있음. 일상생활에서 사용하는 표기 식
- 전위 표기 식 : 연산자가 피연산자 ( )에 있음.
- 후위 표기 식 : 연산자가 피연산자 ( )에 있음.

14.

< 'A+B*C'를 후위 표기 식으로 변환하는 과정 >

① ( )를 기준으로 괄호로 묶어준다.
➜ ( )
② 이 수식에서 우선순위가 높은 연산자는 ( )이므로 먼저 괄호의 ( ) 밖으로 이동시킨다.
➜ ( )
③ 마지막 연산자인 ( )를 괄호의 ( ) 밖으로 이동시킨다. ➜ ( )
④ ( )를 제거한다. ➜ ( )

15.

< 'X+Y/Z'를 후위 표기 식으로 변환하는 과정 >

① ( )를 기준으로 괄호로 묶어준다.
➜ ( )
② 이 수식에서 우선순위가 높은 연산자는 ( )이므로 먼저 괄호의 ( ) 밖으로 이동시킨다.
➜ ( )
③ 마지막 연산자인 ( )를 괄호의 ( ) 밖으로 이동시킨다. ➜ ( )
④ ( )를 제거한다. ➜ ( )

16.

< 연산자들의 스택에서 나오는 과정 >

- ( )에 보관되거나 ( )로 나올 때는 우선 현재 ( )에 있는 연산자와 비교되는 과정이 필요.
- ( )의 연산 **>** ( ) **>** (덧셈과 뺄셈)
- 스택 톱에 있는 연산자의 우선순위가 보관되려는 연산자의 순위보다 ( ) 스택 톱에 있는 연산자가 먼저 스택에서 나옴
- 보관되려는 연산자의 우선순위가 더 ( ) 경우에는 그 연산자가 먼저 스택에서 나옴
- 더이상 보관되려는 연산자가 없는 경우에는 스택에 남아있는 연산자들이 ( )에서부터 나오게 됨.

***서술형 문제**

※ 다음 글을 읽고, 물음에 답하시오.

컴퓨터에서는 데이터를 처리하기 위해 여러 가지 데이터 구조를 활용한다. 그 대표적인 것으로 '스택(stack)'을 들 수 있다. 스택은 한쪽 끝에서만 데이터가 들어가거나 나올 수 있는 특성을 가지는데, 스택에서 데이터가 가장 먼저 나올 수 있는 위치를 톱(top)이라 한다. 스택의 구조는 위쪽만 뚫려 있는 통 안에 책을 차곡차곡 쌓는다고 생각하면 쉽게 이해할 수 있다. 만약 n개의 책을 통에 쌓았다면 가장 먼저 꺼낼 수 있는 책은 아래에서부터 n번째에 있는 것이고, 그 위에 책을 또 올리게 되면 그 책은 아래에서부터 n+1번째에 있게 된다. 스택 구조는 이와 같은 통에, 톱은 가장 위에 있는 책의 위치에 비유될 수 있다. 그리고 스택 알고리즘은 데이터가 들어가 보관되고 나오는 규칙으로, 통에 책을 넣어 쌓고 책을 꺼내는 행위에 비유될 수 있다.

컴퓨터 내의 데이터 처리에서 스택 알고리즘은 수식 변환의 과정에 활용된다. 수식은 '+'(덧셈), '*'(곱 셈), '-'(뺄셈), '/'(나눗셈) 등과 같은 연산자와, 'x, y, z, A, B, C' 등과 같은 변수나 '1, 2, 3' 등과 같은 상수가 포함되는 피연산자로 구성된다. 그리고 수식은 크게 세 가지 표기 식으로 표현할 수 있는데, 연산자가 피연산자 가운데 있는 중위 표기 식, 연산자가 피연산자 앞에 있는 전위 표기 식, 연산자가 피연산자 뒤에 있는 후위 표기 식이 그것이다. 예를 들어 'x 더하기 y 나누기 z'는 중위 표기 식으로는 'x +y/z'로, 전위 표기 식으로는 '+x/yz'로, 후위 표기 식으로는 'xyz/ +'로 표현된다. 우리가 일상생활에서 사용하는 것은 중위 표기 식이지만, 컴퓨터 내부에서는 괄호를 사용하지 않고도 계산해야 할 순서를 알 수 있는 후위 표기 식을 활용하기 때문에 중위 표기 식으로 입력한 수식은 스택 알고리즘을 통해 후위 표기 식으로 변환되어 계산된다.

중위 표기 식을 후위 표기 식으로 변환하는 방법은 간단하다. 기본적인 전제는 피연산자들의 순서는 변하지 않는다는 것이다. 즉 중위 표기 식을 연산자 기준으로 괄호로 묶은 다음, 각 연산자를 묶고 있는 괄호의 오른쪽 괄호 밖으로 연산자를 이동한 후 괄호를 모두 제거하면 된다. 예를 들어 중위 표기 식 'A + B*C'를 후위 표기 식으로 바꾸기 위해서는 '(A +(B*C))'로 괄호를 묶은 뒤, '(A(BC)*) +'와 같이 괄호의 오른쪽에 연산자를 옮긴 후 'ABC* +'와 같이 괄호를 제거하면 된다. 중위 표기 식을 전위 표기 식으로 변환하는 과정 역시 동일하다. 다만 연산자를 오른쪽에 옮기는 것이 아니라 왼쪽으로 옮기는 것만 차이가 있다.

컴퓨터에서는 중위 표기 식을 후위 표기 식으로 변환하기 위해 앞에서 언급한 스택 알고리즘을 활용한다. 후위 표기 식에서 피연산자들의 순서는 변하지 않으므로, 선후 순서가 바뀌어야 되는 연산자들만 다른 위치로 이동하면 된다. 스택 알고리즘에서 피연산자는 순서가 되면 후위 표기로 바로 나오게 된다. 그러나 연산자의 경우 피연산자와 달리 뒤에 스택으로 들어갈 연산자의 존재 여부나 우선순위에 따라 스택에 보관되거나 후위 표기로 나오는 것이 결정된다. 이때 연산자의 우선순위는 괄호 안의 연산을 우선하되, 덧셈과 뺄셈보다 곱셈과 나눗셈을 우선하는 것이 원칙이다. 예를 들어 컴퓨터에서 스택 알고리즘을 활용해 중위 표기 식 'A+B*C'를 후위 표기 식으로 바꾸는 경우, 들어가는 데이터는 'A', '+', 'B', '*', 'C'로 모두 5개이다. 들어가는 5개의 데이터 중 피연산자들은 스택에 보관되지 않고 후위 표기에 순차적으로 나오고, 연산자들은 스택에 순차적으로 보관된다. 다만 연산자들이 스택에 보관되거나 후위 표기로 나올 때에는 현재의 스택 톱에 있는 연산자와 우선순위가 비교되는 과정을 먼저 거친다. 만일 스택 톱에 있는 연산자의 우선순위가 보관되려는 연산자의 우선순위보다 높거나 같으면 스택 톱에 있는 연산자가 먼저 스택에서 나오게 된다. 그런 뒤에는 다시 스택 톱에 오게 되는 다른 연산자와 우선순위가 비교된다. 중위 표기를 다 읽고 나면 연산자가 존재하지 않아 스택에 남아 있는 모든 연산자가 톱에서부터 하나씩 나오게 된다.

1. 스택 알고리즘의 개념을 서술하시오.

2. 일상생활에서 사용하는 중위 표기 식을 컴퓨터 내부에서 후위 표기 식으로 변환하여 사용하는 이유를 서술하시오.

3. 중위 표기 식을 후위 표기 식으로 변환할 때의 기본 전제가 무엇인지 서술하시오.

4. 전위 표기 식과 후위 표기 식의 차이점을 서술하시오.

<조 건>

• 연산자의 위치를 기준으로 서술할 것.

5. 중위 표기 식을 전위 표기 식이나 후위 표기 식으로 변환할 때 연산자와 피연산자의 차이점을 서술하시오.

6. 다음 주어진 중위 표기 식을 후위 표기 식으로 변환하는 과정을 서술하시오.

(A + B) * C

***엮어 읽기 문제**

1. 컴퓨터 알고리즘의 그래프에서 간선은 대상이나 개체를 나타내고, 정점은 이들 간의 관계를 나타낸다.

(O / X)

2. 컴퓨터 알고리즘의 그래프에서 유향 그래프는 상호 간의 친밀도를 표시하는 것처럼 방향성이 필요한 그래프다.

(O / X)

3. 인접 행렬법에서는 가중치의 존재 여부에 상관없이 정점과 정점 사이에 간선이 있으면 행렬의 원솟값을 1로 할당한다.

(O / X)

4. 방향과 가중치가 있는 경우 인접 행렬법으로 표현하면 정점 사이의 인접 여부를 즉각 알 수 있으나, 시간이 많이 든다는 단점이 있다.

(O / X)

5. 무향 그래프를 인접 리스트로 나타내는 경우 간선 하나에 두 개의 노드가 형성된다.

(O / X)

6. 인접 행렬법과 인접 리스트법 모두 존재하지 않는 간선까지 나타내야 한다는 점에서 시간과 공간이 낭비되는 단점을 가진다.

(O / X)

7. 정점과 정점 사이를 잇는 간선에 가중치를 부여할 수 있는데, 이 가중치는 유향 그래프에서만 나타난다.

(O / X)

8. 인접 행렬법에서는 정점의 개수가 10개일 경우, 10X10 행렬이 필요하다.

(O / X)

9. 인접 리스트법에서는 간선 하나에 두 개의 노드가 형성되므로 간선이 10개일 경우 총 20개의 노드가 생성된다.

(O / X)

10. 인접 행렬법은 존재하지 않는 간선까지 나타내야 하므로 인접 리스트법에 비해 시간이 많이 소요된다.

(O / X)

11. 사람들 사이에 친밀도를 나타내는 경우, 각 사람은 하나의 (정점)이 되고, 친밀한 사람들 사이에는 (간선)을 둔다. 또한 (가중치)를 표시하여 친밀한 정도를 나타낼 수 있다. 이것을 인접 행렬법으로 나타내는 경우 (사람 수 X 사람 수)의 행렬이 필요하고, 이 행렬을 통해 사람과 사람 사이의 (인접 여부)를 즉각 알 수 있다.

12. 인접 행렬법의 단점을 서술하시오.

<조 건>

• 인접 리스트법과 비교하여 서술할 것.

정답 및 해설

***지문 확인 문제**

1. <답> 스택, 톱

2. <답> 들어가 보관되고 나오는, 통에 책을 넣어 쌓고 꺼내는

3. <답> 수식 변환, 연산자, 피연산자

4. <답> 2, 3, 중위

5. <답> 중위, 후위, 괄호, 계산해야 할 순서

6. <답> 연산자, 괄호, 오른쪽, 왼쪽

7. <답> 스택 알고리즘, 피연산자들, 연산자들

8. <답> 연산자, 우선순위, 스택, 후위 표기

9. <답> 괄호 안, 덧셈과 뺄셈, 나눗셈과 곱셈, 스택 톱, 스택

10. <답> ①A, B, C ② +, * ③ *

11. <답> 중위, 연산자, 스택, 스택 톱

12. <답> 데이터 구조, 한쪽, 먼저
13. <답> 가운데, 앞, 뒤

14. <답> 연산자, (A+(B*C)), *, 오른쪽, (A+(BC)*), +, 오른쪽, (A(BC)*)+, 괄호, ABC*+

15. <답> 연산자, (X+(Y/Z)), /, 왼쪽, (X+/(YZ)), +, 왼쪽, +(X/(YZ)), 괄호, +X/YZ

16. <답> 스택, 후위 표기, 스택 톱, 괄호 안, 나눗셈과 곱셈, 덧셈과 뺄셈, 높거나 같으면, 높은, 스택 톱

***서술형 문제**

1. <답> 데이터가 들어가 보관되고 나오는 규칙을 스택 알고리즘이라고 한다.

2. <답> 후위 표기 식은 괄호를 사용하지 않고도 계산해야 할 순서를 알 수 있기 때문이다.

3. <답> 피연산자들의 순서는 변하지 않는다.

4. <답> 전위 표기 식은 연산자가 피연산자의 앞쪽에 놓여야 하므로 변환할 때 괄호의 왼쪽 밖으로 이동하고, 후위 표기 식은 연산자가 피연산자의 뒤쪽에 놓여야 하므로 변환할 때 괄호의 오른쪽 밖으로 이동하게 된다.

5. <답> 전위 표기 식이나 후위 표기 식으로 변환할 때, 피연산자들의 순서는 변하지 않지만, 연산자들은 우선순위에 따라 선후 순서가 변한다.

6. <답> 연산자를 기준으로 ((A+B)*C) 묶은 후, 연산자 우선순위에 따라 괄호 안의 연산 '+'를 먼저 괄호의 오른쪽 밖으로 이동하여 ((AB)+*C). 그 다음 마지막 연산자 '*'를 괄호의 오른쪽 밖으로 이동하면 ((AB)+C)*. 끝으로 괄호를 제거하면 AB+C*가 된다.

***엮어 읽기 문제**

1. <답> X
정점이 대상이나 개체를 나타내고, 간선이 이들 간의 관계를 나타낸다.

2. <답> O

3. <답> X
가중치가 있는 경우에는 1이 아닌 가중치 값을 할당한다.

4. <답> O

5. <답> O

6. <답> X
인접 리스트법은 존재하지 않는 간선은 나타내지 않으므로 공간의 낭비가 없다. 하지만 인접 행렬법과 마찬가지로 시간이 많이 소요된다.

7. <답> X
가중치는 무향 그래프에서도 나타난다.

8. <답> O

9. <답> O

10. <답> X
인접 리스트법은 정점과 정점 사이에 간선이 존재하는지를 알아볼 때 리스트를 차례로 훑어야 하므로 인접 행렬법에 비해 시간이 많이 소요된다.

11. <답> 정점, 간선, 가중치, 사람 수X사람 수,
인접 여부

12. <답> 인접 리스트법은 존재하지 않는 간선은 나타나지 않으므로 공간의 낭비가 없으나. 인접 행렬법은 존재하지 않는 간선까지 나타내며 정점의 개수가 n개이면 nXn에 해당하는 공간이 필요해 공간이 많이 낭비된다.