● **Stack는 어떤 자료구조인가?**

▶ 스택은 시간 순서상 가장 나중에, 혹은 최근에 추가한 데이터가 가장 먼저 나오는 후입선출, Last in First Out(LIFO) 형식으로 데이터를 저장하는 자료 구조입니다. 시간복잡도는 push와 pop 둘 다 O(1) 입니다. 활용 예시는 괄호 유효성 검사, 웹 브라우저 방문기록(뒤로 가기), 후위 표기법 연산, 깊이우선탐색(DFS) 등이 있습니다.

● **push & pop**

▶ stack에서 데이터를 추가하는 것을 push라고 하고 데이터를 추출 하는 것은 pop이라고 합니다. push의 경우 stack의 맨 뒤에 데이터를 추가하면 완료되기 때문에 시간복잡도는 O(1)입니다. 이와 동일하게 pop의 경우도 맨 뒤의 데이터를 삭제하면 완료 되기 때문에 O(1)의 시간복잡도를 갖습니다. push와 pop은 모두 stack의 top에 원소를 추가하거나 삭제하는 형식으로 구현됩니다.

● **활용**

▶stack은 재귀적인 특징이 있어서 프로그램을 개발할 때 자주 쓰이는 자료구조입니다. 활용 예시로는 괄호 유효성 검사, 웹 브라우저 방문기록(뒤로가기), 후위 표기법 연산, 깊이우선탐색(DFS), call stack 등이 있습니다.

→ 괄호 유효성 검사: 수식을 왼쪽부터 한 글자씩 읽어서 여는 괄호를 만나면 스택에 push하고, 닫는 괄호를 만나면 스택에서 pop을 해서 수식을 끝까지 검사한 후에 스택이 비어 있는지 확인하는 것으로 올바든 수식인지 검사하는 방법입니다. 검사 과정에서 닫는 괄호를 만날 때 스택에서 가장 나중에 push한 여는 괄호부터 pop하기 때문에, stack의 활용 예시입니다.

→ 웹 브라우저 방문기록(뒤로 가기): 웹브라우져에서 뒤로 가기 버든을 누르면 가장 나중에 연 페이지부터 다시 열어서 보여 줍니다. LIFO으로, stack의 활용 예시입니다. 비슷한 예시로는 실행취소(control + z)가 있으며, 이 버든은 가장 나중에 실행된 것부터 실행을 취소합니다.

→ 후위 표기법 연산: 연산자를 피연산자의 뒤에 놓는 방법입니다. 중위표기를 후위표기로 바꾸는 알고리즘에서 연산자는 스택의 맨 위에 있는 앞 연산자와 비교해서 출력하거나 스택에 넣어 대기시키는데, 이때 대기 된 연산자 중에서는 가장 나중에 대기 된 연산자가 가장 먼저 나오며 처리됩니다. LIFO, 따라서 스택을 이용한 예시입니다.

→ 깊이우선탐색(DFS): 노드를 방문 순서 대로 스택에 저장해나가다가 가장 나중에 방문처리한 노드에 더이상 인접 노드가 없거나 인접 노드를 모두 방문했으면 해당 노드를 스택에서 내보내는데, 이때 나중에 방문처리한 노드부터 처리해나가기 때문에, LIFO으로 Stack의 활용 예시입니다.

Resources: inflearn 개발남 노씨