

1. 열거형에 ASCII 코드 값을 할당하여 바로 이용할 수 있게 하였다.
2. 스택을 연결리스트로 구현하였다.
3. infix와 postfix를 각각 배열로 수집했다.
4. 파일의 총 길이를 구하여 그만큼을 최대길이로 하여 infix, postfix를 동적 할당했다. postfix의 경우에는 공백을 추가하기 위해 파일 길이의 두 배로 넉넉하게 잡았다.
5. infix를 수집할 때 편의성과 안정성을 위해 공백을 모두 제거하였다.
6. 중요도를 나타내는 함수를 만든다. 이 때 앞서 스택에 있는 연산자의 중요도와 새로 심사 중인 연산자의 중요도 기준을 다르게 만든다. top이 NULL일 경우에는 -1을 리턴한다.
7. 스택의 top이 가지고 있는 값을 출력하는 함수와 top을 삭제하는 함수, 새로운 top을 추가하는 함수를 구현한다.
8. infix를 하나씩 읽는 함수를 만들었다. 연산자의 경우에는 연산자를 곧바로 출력하고, 연산자가 아닐 경우에는 피연산자임을 알려준다.
9. 읽어 들인 infix 값이 연산자일 경우에는 토큰에 할당한다. 피연산자일 경우에는 postfix에 추가한다.
10. 토큰의 중요도가 앞서 스택에 저장된 값의 중요도보다 크거나 같은 경우에는 토큰을 스택에 추가한다. 만약 작은 경우에는 우선 top의 값이 ‘(‘인지, NULL인지 검사한 후, 둘 다 아니라면 둘 중 하나라도 거짓이 될 때까지 스택의 값을 postfix에 추가한다. 끝난 뒤. top이 ‘(‘일 때, 토큰이 ‘)’라면 둘 다 지우고 다음 단계로 넘어가며, 그렇지 않고 전자만 만족한다면, ‘(‘을 남겨두고 토큰을 스택에 저장한다. 만약 top이 NULL이라면 토큰을 스택에 저장한다. 토큰이 ‘\0’가 된다면 스택에 남은 연산자들이 모두 postfix에 들어가게 되며 ‘\0’까지 마지막으로 들어가게 된다.
11. postfix에 값을 집어넣는 경우에는 가독성을 위해 공백을 추가한다.
12. 마지막으로 postfix의 내용물을 출력하며 프로그램이 끝난다.
13. 첫 줄은 원본 텍스트, 두 번째 줄은 공백을 제거한 infix, 세 번째줄은 결과물인 postfix이다.