Internal document

enum TokenType{}: 열거형으로 정의한 TokenType은 소스 코드에서 사용되는 다양한 토큰 유형을 나타내는 식별자를 정의한다. 각 토큰 유형은 소스 코드의 구문 요소를 분류하는 데 사용됩니다. 예를 들어, 식별자, 상수, 연산자, 세미콜론 등이 있다.

struct Token{}: 이 구조체는 토큰의 유형(TokenType)과 해당 토큰의 문자열 값을 저장한다.

글로벌 변수: 다음 토큰, 토큰 문자열, 입력 파일 스트림, 카운터, 에러 및 경고 플래그 등의 상태를 추적하는 데 사용되는 변수들.

식별자 및 상수 확인 함수 (is_ident, is_const): 주어진 문자열이 식별자나 상수인지를 확인

lexical(): 입력 파일에서 다음 토큰을 읽고, 이를 분류하여 next_token 글로벌 변수에 저장합니다. 공백을 무시하고, 식별자, 상수, 연산자 등을 인식한다.
program(): 프로그램의 시작점으로, statements 함수를 호출하여 프로그램 내의 모든 문장들을 처리한다.
statements(): 여러 개의 statement를 반복적으로 처리한다. 각 문장은 세미콜론으로 구분된다.
statement(): 하나의 문장을 처리한다. 식별자로 시작해야 하며, 할당 연산자와 표현식이 뒤따라야 한다.
expression(): 수학적 표현식을 처리한다. term과 term_tail 함수를 호출하여 표현식을 구성하는 항들을 처리한다.
term(): 표현식 내의 항을 처리한다. factor와 factor_tail 함수를 호출하여 항을 구성하는 요소들을 처리한다.
term_tail(): 항 뒤에 오는 추가적인 항들을 처리한다. 주로 덧셈 또는 뺄셈 연산자를 처리한다.
factor(): 항 내의 요소를 처리한다. 괄호로 묶인 표현식, 식별자, 상수를 인식한다.
factor_tail(): 요소 뒤에 오는 추가적인 요소들을 처리한다. 주로 곱셈 또는 나눗셈 연산자를 처리한다.
evaluate_expression(): 분석된 표현식을 계산하고 그 결과를 정수로 반환한다.
evaluate_term(): 분석된 항을 계산하고 그 결과를 정수로 반환한다.
evaluate_factor(): 분석된 요소를 계산하고 그 결과를 정수로 반환한다.
main(int argc, char* argv[]): 프로그램의 진입점으로, 필요한 인자를 받아 파일을 열고, lexical 함수를 호출하여 첫 번째 토큰을 읽은 후, program 함수를 호출하여 프로그램을 실행하고, 결과를 출력한다.