Internal document

소프트웨어학부 20223908 김주영

Process

<ident>의 현재 값을 저장하기 위한 data-structure로 Vector를 사용하였다. Vector의 자료형은 <pair<string, int>>이다. 여기서 string은 <ident>의 이름을 저장하고, int는 그 변수의 값을 저장한다.

먼저 프로그램을 크게 lexical analyzer부분과 parser로 구분할 수 있다.

Lexical analyzer의 역할은 파일에서 문자단위로 읽어서 읽은 요소들을 합쳐서 lexeme을 구성한다. 그 lexeme에 대한 정보를 각종 변수에 저장해 둔다. (ex. Lexeme, token, charClass, etc.)

다음으로 parser의 역할은 <statement>의 의미를 파악하여 실질적인 연산을 한다. 이 과정에서 예외에 대한 처리도 같이 한다.

Handling exception

오류 처리를 위해 error함수를 선언하였다. 이는 매개변수로 정수형 변수 debug를 전달받는다. 이는 열거형 ERROR_CODE에서 미리 정의된 코드로써 어떤 유형의 에러인지에 대한 정보를 포함한다. error함수는 전달받은 debug 변수를 통해 error_flag 변수를 해당 오류코드로 할당한다.

오류가 발견되어도 파싱을 중단하지 않고 이어서 해야 하기 때문에 error_flag 전역변수를 선언한다. 파싱하는 과정에서 에러가 발견되면 이 변수에 저장해두고 이어서 파싱한다. 파싱이 완료되면 error_flag를 통해 하나의 <statement>를 parse한 결과를 출력할 때 오류가 있을 시 어떤 오류인지 출력할 수 있도록 한다.

처리한 예외는 3가지로 no_declaration, not_enough_elements, wrong_parenthesis가 있다. 이는 각각 "정의되지 않은 변수가 참조됨", "식의 요소가 부족함", "올바르지 않은 괄호쌍"를 의미한다.

정의되지 않은 변수

<statement>의 우항을 계산하는 과정에서 모든 항에 대해서 상수가 아닌 항 중에(변수) 미리 선언되지 않은 항이 있는지 확인한다. 이는 <ident>를 저장하고 있는 ident Vector에서 확인한다. 만약 없을 시 error함수에 1번 에러코드를 전달한다. 또한 result를 출력할 때 결과가 Unknown이라도 <ident>에 대한 정보가 출력되어야 하므로 일단 ident Vector에 추가한다.

식의 요소가 부족함

이 예외의 엄밀한 의미를 연산자, 혹은 피연산자의 개수가 올바르지 못하다는 것이다. 예를 들어 "a + 2 /"와 같은 수식이 있을 때, 이는 단순히 피연산자가 부족하다는 것으로도 cover할 수 있지만 더욱 일반화하여 식의 요소가 부족한 것으로 cover하였다. 우항에 존재하는 operator(<add_operator>,<mult_operator>)의 개수와 operand(<const>,<ident>)의 개수를 비교해서 올바르지 못한 경우 error함수에 2번 코드를 전달한다.

올바르지 않은 괄호

올바르지 않은 괄호를 다음과 같이 정의할 수 있다.

1) 여는 괄호와 닫는 괄호의 개수가 다른 경우

2) 여는 괄호 뒤에 닫는 괄호가 없는 경우

이를 cover하기 위해 여는 괄호의 개수와 닫는 괄호의 개수를 저장한다. 또한 계산하는 과정에서 전체 수식에서 가장 안에 있는 괄호쌍부터 계산하고 전체 수식의 괄호 수식이 있던 부분을 그 결과로 대체한다. 이렇게 반복적으로 괄호가 없어질 때까지 계산한다. 이렇게 계산하기 위해 연속된 괄호쌍을 찾는 과정이 필요하다. (ex. "(()())"에서는 1번 인덱스와 2번 인덱스가 연속으로 여닫는 괄호가 존재하므로 이 부분을 찾음) 이 과정에서 여는 괄호보다 닫는 괄호를 찾는 경우 error함수에 3번 코드를 전달한다.