컴파일러 설계 프로젝트 제안서

프로젝트 제목	테스트케이스 작성에 특화된 언어 설계
설계 프로젝트 개요	**개요** 새로운 프로그래밍 언어를 설계한다. 해당 언어는 형식화된 출력을 보다 편리하게 할 수 있도록 설계된다. Ruby 언어에서 사용되는 "#{}"과 같은 문장을 보다 쉽게 사용할 수 있도록 <, > 와 {, }를 이용하여 출력하거나, C 언어에서 사용되는 printf와 같은 함수를 보다 쉽게 사용할 수 있도록, print 연산자를 이용하여 출력할 수 있다. 또한 모든 연산자(operator)는 왼쪽에 위치하고, 모든 피연산자(operand)는 소괄호 (,)를 이용해서 지정된다. 따라서 프로그래머는 피연산자의 숫자에 따라서 여러 개의 연산자를 사용할 필요가 없이 연산자는 한 개만 기술하는 것이 가능하다. 또한 연산자의 피연산자에 다른 표현식(expression)이 중첩(nested)될 수 있다. 모든 표현식은 값(value)을 가진다. 또한 기본적인 출력은 STDOUT이고, output 연산자를 이용하여 redirection을 할 수 있고, 파일에 출력하는 것도 쉽게 할 수 있다. 이 언어를 이용하면 간단한 숫자 리스트 또는 특정 형식의 리스트 또는 테스트케이스와 같은 리스트의 생성을 편리하게 할 수 있다. **목적 및 필요성** 간단하게 계산된 값을 특정 형식에 맞춰 출력을 해야 할 때, 기존의 C언어나 다른 언어를 이용하는 경우 간단한 내용이지만 해당 언어의 복잡한 수식을 이해하거나 복잡한 출력형식을 사용해야 하는 경우가 많이 있다. 계산된 값이나 출력형식은 간단하지만 해당 내용을 출력하기 위해서는 언어마다 다른 복잡한 I/O를 이해해야 한다. 이러한 복잡한 I/O를 언어에서 처리하여 프로그래머는 간단한 방식으로 특정 형식에 맞춰 출력을 할 수 있게 한다. 또한 피연산자가 2개 이상인 경우 중위표현을 사용하는 언어에서는 연산자가 1개 이상 필요하다. 하지만 모든 표현식을 전위표현을 사용하여 프로그래머는 피연산자의 개수에 상관없이 연산자를 한 개만 사용하면 된다. 또한 연산자를 중첩하여 사용할 수 있으므로 모든 수식을 표현하는데 문제가 없다. **현실적 제한요소** 다양한 I/O를 보다 단순하게 추상화하여 처리가 복잡하다. 또한 I/O 관련 에러는 까다롭고 매우 다양하므로 관련된 모든 에러를 처리하는 것은 현실적인 한계가 있다. 또한 표현식의 중첩에 따른 계산은 계속해서 재귀적으로 중첩된 표현식을 계산하므로, 너무 많은 중첩된 표현식을 계산하는 경우 내부적으로 에러를 발생시킬 여지가 매우 크다. **기대효과:** 다양한 경우에 간단한 숫자 리스트 또는 특정 형식의 리스트로 구성된 파일 또는 출력이 필요하다. 이런 경우 해당 형식으로 된 리스트를 생성하기 위해서 기존의 언어를 사용하면 복잡한 I/O 방식을 이해하고 처리해야 한다. 이런 경우에 실제 원하는 리스트를 생성하는 알고리즘은 간단하지만 반대로 출력하거나 특정 파일에 저장하는 연산은 복잡한 경우가 많다. 따라서 프로그래머가 테스트케이스를 생성하는 일과 같은 반복적이고 간단한 리스트를 생성하는

작업을 보다 쉽고 간단하게 프로그래밍 할 수 있도록 할 수 있게 해준다.

결론

프로그래밍을 하는 경우에 실제 문제를 해결하기 위한 프로그램의 작성을 하는 경우도 많이 있지만 해당 프로그램이 실제로 해당 문제를 정확히 해결하는지 테스트 하기 위해서는 다양한 테스트케이스가 필요하다. 사람이 특정 테스트케이스를 생성하는 경우도 있지만, 보다 다양하고 많은 테스트를 위해선 자동화된 테스트케이스를 생성해야 한다. 이때 기존의 언어를 사용하면, 테스트케이스를 생성하는 알고리즘은 간단하나 해당 테스트케이스를 생성하는 과정은 복잡한 경우가 많다. 특히나 I/O 관련된 처리는 언어마다 다르지만 복잡한 경우가 대부분이다. 따라서 이러한 I/O 관련 처리에 특화하고, 표현식을 단순화하여 프로그래머가 테스트케이스의 생성을 보다 쉽게 할 수 있다.

(참고) 2페이지 이내로 작성. 현실적 제한요소는 구현의 한계, 실용화 문제점, 제약 요소 등을 기술.