Principles of Programming Languages

Autumn, 2025

Programming Assignment

< 문 제 >

※ 파일을 통해 입력된 프로그램이 아래에 제시된 LL(1) 문법을 따르는지 여부를 판단하는 파서(Parser)를 Recursive Descent Parsing 기법을 이용하여 작성하고, 파싱된 결과를 이용하여 입력된 프로그램의 결과를 출력하시오.

< 문 법 >

```
<program> → <statements>
<statements> → <statement> <stmts_tail>
<stmts_tail> → <semi_colon> <statement> <stmts_tail> | ε
<statement> → <ident> <assignment_op> <expression>
<expression> → <term> <term_tail>
<term_tail> → <add_op> <term> <term_tail> | ε
<term> → <factor> <factor_tail>
<factor_tail> → <mult_op> <factor> <factor_tail> | ε
<factor> → <left_paren> <expression> <right_paren>
         | <ident> | <const>
<const> → num // any decimal number literal[1)]
<ident> → id // C identifier token[2)]
<assignment_op> → :=
<semi_colon> → ;
<add_operator> → + | -
<mult_operator> → * | /
<left_paren> → (
<right_paren> → )
```

< 세부사항 >

※ 개발 조건

① C/C++/C#/Java/Python으로 개발된 **Command Line Application**만 허용
(오픈소스기반 GCC, Python 추천, C/C++/C#/Java의 경우 10% 가산점, 예) 점수 100 -> 110, 90 -> 108)

② 자신(혹은 팀)의 학번으로 명명된 폴더 내에 작성된 모든 소스 코드 저장
예) 20240000_손흥민_20240001_이강인

③ Java의 경우, 폴더 최상단에 Main.jar를 Runnable Jar 파일로 생성한 다음, 'java –jar Main.jar'를 실행

④ Python의 경우, main.py을 진입점(entry point)으로 하고 'python main.py'를 실행

※ 주의: 제출한 프로그램이 컴파일되지 않으면 평가 대상에서 제외됩니다. 따라서, 외부문서(External Document)에는 프로그램 실행에 필요한 구성요소와 컴파일 및 실행 방법을 반드시 구체적으로 설명해야 합니다.

※ 입력: 임의의 이름이 부여된 텍스트 파일 (Command Line 파라미터로 파일명이 주어짐.
(예: "python main.py **eval1.txt**")
(예: "a.out **eval1.txt**")

※ 출력: 주어진 문법에 따라 입력파일에 저장되어 있는 프로그램을 분석한다.
파싱(parsing)되는 과정을 **<처리 예>**와 같이 출력하고, 문법 오류 없이 파싱된 경우, **파싱 트리를 구축**하고, **이 트리를 이용하여 <ident>들의 최종값을 출력**한다.
=> 출력 형식:
"프로그램에서 읽은 라인"
"ID: {개수}; CONST: {개수}; OP: {개수};"
"파싱 결과 (OK), (ERROR)"
"Result ==> {변수1}: {최종값}; {변수2}: {최종값}; {변수3}: {결과값};"

※ 처리 조건

각 문장들이 파싱된 이후, 입력된 문장과 그 문장에 포함된 식별자(IDENT), 숫자(CONST), 그리고 연산자(OP)의 개수를 출력한다.
파싱된 문장이 문법에 적합하면 '(OK)', 적합하지 않으면 적절한 에러 '(ERROR)' 메시지를 출력한다.

위에서 주어진 문법에 의거 오류가 발견된 경우, 에러 '(ERROR) 메시지'를 출력하고 파싱을 계속하되, 이 경우 해당 식별자(<IDENT>)의 값은 'Unknown'으로 결정된다. 문장이 포함하고 있는 모든 오류에 대해 에러 메시지를 출력해야 한다. 에러메시지 내용은 각자 적절히 정의한다(**문서에 표기해야 함**). 프로그램에 속한 일부 문장이 문법에 적합하지 않더라도 프로그램이 끝까지 전부 파싱되어야 한다.

파싱 트리 생성 후, 모든 <IDENT> 값이 출력되어야 한다. 단, <IDENT>의 값이 정의되지 않은 경우, "Unknown"으로 표시한다.
<IDENT>의 현재 값을 저장하기 위해 심볼 테이블(symbol table)을 구축해야 한다.
입력 스트림에서 ASCII 코드값이 32 이하인 것은 모두 white-space로 간주되며, white-space는 각 token을 구별하는 용도 이외에는 모두 무시된다.

어휘분석기(lexical analyzer)의 소스 코드는 정수 변수 next_token, 문자열 변수 token_string, 함수 lexical()을 포함하여야 한다. 함수 lexical()은 입력 스트림을 분석하여 하나의 lexeme을 찾아낸 뒤, 그것의 token type을 next_token에 저장하고, lexeme 문자열을 token_string에 저장하는 함수이다. 기타 구현 시 요구되는 세부 사항은 직접 결정하고, Internal 및 External Document에 기술한다.

1) Decimal Numbers (십진수)

- 기수(base)가 10인 숫자 체계, 정수나 실수 포함.

2) A set of rules to create an identifier (식별자) in C

- 식별자는 다음 문자를 포함할 수 있다.
  - Uppercase (A-Z) and lowercase (a-z) alphabets
  - Numeric digits (0-9)
  - Underscore (_)
- 식별자의 첫 글자는 Letter(문자)나 underscore(밑줄)만 가능하다
- 식별자는 case-sensitive이다.
- 식별자는 C언어의 키워드(예: int, return, …)가 될 수 없다.

< 처리 예 >

※ 입력 #1

operand1 := 3 ;

operand2 := operand1 + 2 ;

target := operand1 + operand2 * 3

※ 출력 #1

operand1 := 3;

ID: 1; CONST: 1; OP: 0;

(OK)

operand2 := operand1 + 2;

ID: 2; CONST: 1; OP: 1;

(OK)

target := operand1 + operand2 * 3

ID: 3; CONST: 1; OP: 2;

(OK)

Result ==> operand1: 3; operand2: 5; target: 18;

※ 입력 #2

operand2 := operand1 + 2 ;

target := operand1 + operand2 * 3

※ 출력 #2

operand2 := operand1 + 2;

ID: 2; CONST: 1; OP: 1;

(Error) “정의되지 않은 변수(operand1)가 참조됨”

target := operand1 + operand2 * 3

ID: 3; CONST: 1; OP: 2

(OK)

Result ==> operand1: Unknown; operand2: Unknown; target: Unknown

※ 입력 #3

operand1 := 1;

operand2 := (operand1 * 3) + 2 ;

target := operand1 + operand2 * 3

※ 출력 #3

operand1 := 1;

ID: 1; CONST: 1; OP: 0

(OK)

operand2 := (operand1 * 3) + 2;

ID: 2; CONST: 2; OP:2;

(OK)

target := operand1 + operand2 * 3;

ID: 3; CONST: 1; OP: 2;

(OK)

Result ==> operand1: 1; operand2: 5; target: 16

※ 입력 #4

operand1 := 3 ;

operand2 := operand1 + + 2 ;

target := operand1 + operand2 * 3

※ 출력 #4

operand1 := 3;

ID: 1; CONST: 1; OP: 0;

(OK)

operand2 := operand1 + + 2;

ID: 2; CONST: 1; OP: 1;

(ERROR) “중복 연산자(+) 발생”

target := operand1 + operand2 * 3

ID: 3; CONST: 1; OP: 2;

(OK)

Result ==> operand1: 3; operand2: Unknown; target: Unknown

< 제출 관련 사항 >

제출물

- Internal / External Documents **(pdf 형식)**
  - Internal Document: 작성한 소스코드를 간략히 설명 (어느 부분에서 무엇을 작업하는 지(각 함수들의 역할 설명) 정도 요약)
  - External Document: 프로그램의 실행 방법 및 요구사항 자세히 설명바람 (이 문서를 보고 제출한 소스코드를 컴파일하고, 실행방법을 따라서 수행할 예정)
- 프로그램 소스 코드 (실행파일 제외, External Document에서 제공하는 실행 방법을 통해 소스 코드를 직접 실행 또는 컴파일 후 실행)

제출 방법

- 모든 제출물을 압축하여 **학번1_이름1_학번2_이름2.zip** 형식으로 eclass에 제출

제출 마감 일시

- **2025년 11월 12일 (수) 오후 11:59** (Late Penalty: -5%/Day)