**PL 파서 프로젝트**

**1. 주요 기능 설명**

**(1) 토큰(Token) 및 문법 요소**

* 소스 코드에는 PL 언어에서 사용하는 다음의 주요 토큰 및 연산자들이 정의되어 있습니다:
  + **토큰 타입(TokenType):**
    - assignOp : 할당 연산자 (:=)
    - addOperator : 덧셈/뺄셈 연산자 (+, -)
    - multOperator : 곱셈/나눗셈 연산자 (\*, /)
    - semiColon : 세미콜론 (;)
    - ident : 식별자
    - constant : 상수
    - leftParen / rightParen : 왼쪽/오른쪽 괄호 ((, ))
    - unknownToken : 알 수 없는 토큰
    - endOfFile : 파일 종료

**(2) 파싱 흐름**

1. **파일 읽기 및 처리**:
   * main 함수는 입력 파일 경로를 받아 파일 내용을 한 줄씩 읽습니다.
   * 줄을 처리하며 문장을 파싱하고 결과를 출력합니다.
2. **토큰화(Tokenization)**:
   * 함수 getToken이 입력 줄에서 토큰을 추출하고 토큰 배열(tokenArray)에 저장합니다.
   * 공백, 연산자, 괄호, 상수, 식별자 등을 구분하여 각 토큰을 생성합니다.
3. **문장 파싱(Parse Statements)**:
   * parseStatements와 parseStatement는 문장 단위로 파싱을 수행합니다.
   * 할당 연산자를 통해 변수와 값을 매칭하거나, 식(Expression)을 파싱하여 값을 계산합니다.
4. **식(Expression) 파싱**:
   * parseExpression, parseTerm, parseFactor 등의 함수는 연산자 우선순위에 따라 수식을 재귀적으로 분석합니다.
   * 괄호로 묶인 식, 변수 참조, 상수 등 다양한 경우를 처리합니다.
5. **식별자 값 관리**:
   * 식별자와 그 값을 저장 및 조회하는 기능을 setIdentValue, getIdentValue 함수로 제공합니다.
   * 정의되지 않은 식별자 사용 시 에러를 출력합니다.

**(3) 출력 기능**

* printResultByLine: ID(식별자), 상수, 연산자 수를 출력합니다.
* printIdent: 모든 식별자의 값을 출력합니다.
* printOPWarning: 잘못된 연산자 사용에 대한 경고 메시지를 출력합니다.
* printIDError: 정의되지 않은 식별자 사용에 대한 에러 메시지를 출력합니다.
* printOK: 현재 줄이 문법적으로 올바르면 "(OK)"를 출력합니다.
* printToken: -v 모드에서 각 토큰의 종류를 출력합니다 (예: assignment\_op, ident).

**2. 주요 함수, 데이터 구조 및 변수**

1. **TokenType 열거형**

토큰의 종류를 정의하는 열거형으로, 각 토큰의 역할에 따라 구분됩니다.

* **assignOp**: 할당 연산자 (:=)
* **addOperator**: 덧셈/뺄셈 연산자 (+, -)
* **multOperator**: 곱셈/나눗셈 연산자 (\*, /)
* **semiColon**: 세미콜론 (;)
* **ident**: 식별자 (예: 변수명)
* **constant**: 상수 (숫자값)
* **leftParen**: 왼쪽 괄호 (()
* **rightParen**: 오른쪽 괄호 ())
* **unknownToken**: 알 수 없는 토큰
* **endOfFile**: 파일 종료를 나타냄

1. **주요 변수 설명**

| **변수명** | **설명** |
| --- | --- |
| line | 현재 입력 줄을 저장하는 문자열 배열. |
| file | 입력 파일의 파일 포인터. |
| idArray | Ident 구조체 배열로, 식별자 이름과 그 값을 저장. |
| addPosition | 현재 읽고 있는 토큰 문자열의 끝 위치를 나타냄. |
| getPosition | 현재 읽고 있는 입력 줄에서의 위치. |
| pushPosition | 저장된 토큰 배열에서 다음 저장 위치를 나타냄. |
| idCount | 현재 줄에서 발견된 식별자 수. |
| constCount | 현재 줄에서 발견된 상수 수. |
| opCount | 현재 줄에서 발견된 연산자 수. |
| totalId | 전체 코드에서 발견된 식별자 수. |
| idErrorFlag | 정의되지 않은 식별자 에러가 발생했는지 여부를 나타냄. |
| opWarningFlag | 중복 연산자와 같은 경고 발생 여부를 나타냄. |
| verboseFlag | -v 옵션 활성화 여부를 나타냄. |
| currentChar | 현재 읽고 있는 문자. |
| tokenString | 현재 읽고 있는 토큰 문자열. |
| errorIdent | 에러가 발생한 식별자의 이름. |
| nextToken | 다음으로 처리할 토큰 정보. |
| tokenArray | 현재 줄에서 추출된 토큰 배열. |

1. **주요 함수 설명**

| **함수 이름** | **설명** |
| --- | --- |
| **main** | 입력 파일을 읽고 줄 단위로 처리하며, -v 옵션에 따라 동작 변경. |
| **parse** | 일반 모드에서 토큰을 분석하고 파싱하여 결과 출력. |
| **parse\_V** | -v 모드에서 각 토큰을 분석하고 출력. |
| **getToken** | 입력 줄에서 토큰을 추출하고 토큰 배열에 저장. |
| **parseExpression** | 수식(Expression)을 파싱하여 결과 계산. |
| **parseStatements** | 여러 문장(Statements)을 파싱. |
| **setIdentValue** | 식별자와 그 값을 설정. |
| **getIdentValue** | 식별자의 값을 조회. 정의되지 않았을 경우 "Unknown" 설정. |
| **printResultByLine** | 줄 단위로 ID, 상수, 연산자 수를 출력. |
| **printOPWarning** | 잘못된 연산자 사용에 대해 경고 메시지 출력. |
| **printIDError** | 정의되지 않은 식별자 사용 시 에러 메시지 출력. |

**3. 실행 예시**

##### ****예제 1: 기본 파싱****

**입력:**

x := 10;

y := x + 5;

z := (x + y) \* 2;

**출력:**

x := 10;

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

(OK)

y := x + 5;

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

(OK)

z := (x + y) \* 2;

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

(OK)

Result ==> x: 10; y: 15; z: 50;

##### ****예제 2: 정의되지 않은 식별자****

**입력:**

a := b + 10;

**출력:**

a := b + 10;

(Error) "referring to undefined identifiers(b)"

##### ****예제 3: 중복 연산자****

**입력:**

x := 5 + + 3;

y := 10 \* \* 2;

**출력:**

x := 5 + + 3;

(Warning) "Eliminating duplicate operator (+)"

(OK)

y := 10 \* \* 2;

(Warning) "Eliminating duplicate operator (\*)"

(OK)

##### ****예제 4: 잘못된 할당 연산자****

**입력:**

x = 5;

**출력:**

x = 5;

(Warning) "Substituting assignment operator (:=)"

(OK)

##### ****예제 5:**** -v ****모드 (자세히 출력)****

**입력:**

x := 5 + (3 \* y);

**출력:**

assignment\_op

ident

add\_operator

left\_paren

const

mult\_operator

ident

right\_paren

(Error) "referring to undefined identifiers(y)"

##### ****예제 6: 괄호 처리****

**입력:**

z := (x + 5) \* (y + 3);

**출력:**

z := (x + 5) \* (y + 3);

(Error) "referring to undefined identifiers(x)"

(Error) "referring to undefined identifiers(y)"