* RDP class: Recursive Descent Parsing을 수행한다.
  + 토큰 종류를 심볼 종류를 표현하는 열거형 TOKEN\_TYPE으로 정의한다.
  + 생성자와 소멸자 사용해 파일을 열고 닫는다.
* addToken(): 토큰을 토큰 목록에 추가한다.
  + \_tokens 벡터에 현재 tokenString 값을 추가한다.
  + \_tokens 벡터: 파싱된 토큰들을 저장한다.
* getChar(): 다음 문자를 읽어온다.
  + \_ch: 현재 파싱 중인 문자.
* isSpace(), isOperator(), isSpecial(): 문자의 공백 여부, 연산자 여부, 특수 문자 여부를 판별한다.
* msgError(), msgWarning(): 오류, 경고 메시지 생성 및 결과에 메시지를 추가한다.
* printStatement(): 파싱된 문장을 출력한다.
  + \_tokens 벡터를 순회하며 토큰들을 출력한다.
  + \_ID, \_CONST, \_OP 값을 출력해 각각의 개수를 표시한다.
  + \_result에 오류/경고 메시지가 있으면 출력한다.
* printSymbolTable(): 심볼 테이블을 출력한다.
  + \_symTable: 맵을 순회하며 변수 이름과 값을 출력한다.
* factor(): <factor>를 처리한다.
  + nextToken == 식별자: 변수를 처리하고 \_symTable에 해당 변수를 추가한다.
  + nextToken == 상수: 상수를 처리한다.
  + nextToken == ‘(‘: 괄호 안의 표현식을 계산한다.
  + else: 오류를 처리한다.
* term(): <term>을 처리한다.
  + factor()를 호출해서 term을 계산하고 ‘\*(곱셈)’ 또는 ‘/(나눗셈)’을 처리한다.
* expression(): <expression>을 처리한다.
  + term()을 호출해서 expression을 계산하고 ‘+(덧셈)’ 또는 ‘-(뺄셈)’을 처리한다.
* statement(): 각 문장을 처리한다.
  + nextToken을 검사해서 변수 할당문을 처리하고 \_symTable을 업데이트한다.
  + \_result에 오류 메시지를 추가한다.
* statements(): 문장 여러 개를 처리한다.
  + lexical()을 호출해서 다음 토큰을 읽고 statement()를 호출해서 문장을 처리한다.
* lexical(): 토큰을 읽고 종류를 결정한다.
  + 문자를 인식하고 해당하는 nextToken값을 설정한다.
  + 오류 처리도 수행한다.
* main(): 프로그램의 진입점이다.
  + 입력 파일을 열고 파싱 작업 수행한다.

예외 처리

* RHS 시작하는/중간 부분에 연산자가 오는 경우: (Warning) 중복된 토큰을 제거한 뒤 다음 토큰을 불러온다. -> 정상적인 결과값을 출력한다.
* variable이 정의되지 않은 경우: (Error) 문장 자체는 끝까지 처리하지만 해당 Variable은 처리하지 않는다. -> 결과값은 unknown을 출력한다.
* assignment가 잘못된 경우: (Error) 문장 자체는 끝까지 처리하지만 해당 Variable은 처리하지 않는다. -> 결과값은 unknown을 출력한다.
* 한 문장에 Warning 또는 Error가 2개 이상인 경우: 해당되는 모든 Warning/Error 메시지를 출력하도록 한다. -> 결과값은 처리 가능 여부에 따라 달라진다.