Hw1 보고서

201820805 홍성표

1. 서론

|  |  |  |
| --- | --- | --- |
|  | 구현 O | 구현 X |
| ID | 최대 식별 길이 10에 대한 처리  중복에 대한 처리  Abc123a에 대한 오류 처리 |  |
| INT | 00 에 대한 오류 처리 |  |
| REAL | 0.30 에 대한 오류 처리  +.13에 대한 구현  0.13에 대한 구현 |  |
| STRING | \t, \n 에 대한 처리  “\”에 대한 오류처리 |  |
| OPERATION | 연산자 +,-,/,\*,=,: 에 대한 처리 | 괄호에 대한 구현 |
| SEMICOLON | 문장 마지막 “;”이 없는 것에 대한 오류 처리 |  |

1. 문제 분석
   1. Token의 종류
      1. ID - 식별자
      2. INT - 정수
      3. REAL - 실수
      4. STRING - 문자열
      5. OPERATION - 연산자
      6. SEMICOLON - 세미콜론
   2. Transition diagram

텍스트, 화이트보드이(가) 표시된 사진

자동 생성된 설명

1. 설계

While과 fget을 사용해 문자열을 받아오고 입력 또는 파일의 끝이 나올 때까지 반복하여 입력을 받아 분석하였다.

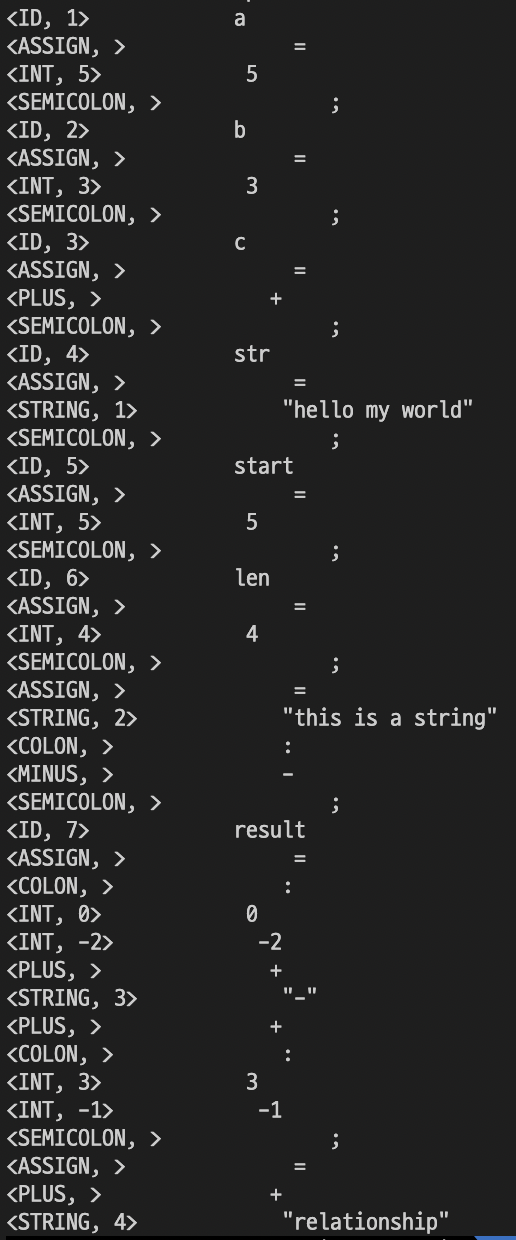
Token이라는 구조체를 사용하여 토큰의 정보를 저장하였다. 구조체 안에는 data, name, code\_num, idx가 있으며 data는 토큰의 문자열 name은 토큰의 종류, code\_num은 토큰이 에러인지 아닌지를 정수형태로 저장(0: error, 1: id, 2: operator, 3: int, 4: float, 5: string, 6: etc), idx에는 symbol table과 string table을 만들기 위한 id와 string의 idx를 저장하였다.

구조체 Token을 저장할 배열을 만들어 fget()을 통해 받아온 문자열을 토큰 단위로 하나씩 분석해 나갈 때 마다 생기는 토큰을 저장했다.

ID의 경우 abc123a가 입력되었을 때 abc123은 오류처리 a는 정상적 id처리를 하였다.

STRING의 경우 “abc\def”가 입력으로 들어왔을 때 “\”만 오류 처리후 “abcdef”는 정상 string처리를 하였다.

1. 입력 데이터
   1. ex1.txt
      1. a = 5;
      2. b = 3;
      3. c = a + b;
      4. str = "hello my world";
      5. start = 5;
      6. len = 4;
      7. str = "this is a string":start-len;
      8. result = str:0-2 + "-" + str:3-1;
      9. result = result + "relationship"
   2. ex2.txt
      1. a = 00; abc123def
      2. b = +0.130;
      3. str = "hello\
      4. my world!";
2. 결과 데이터(화면 캡쳐)
3. ex1.txt

텍스트이(가) 표시된 사진

자동 생성된 설명

1. ex2.txt

텍스트이(가) 표시된 사진

자동 생성된 설명