컴파일러 설계 project1 Report

2017029616 박병현

* Compilation

1) 환경: Ubuntu 20.04

2) 방법: make -> ./cminus\_cimpl <input filename> ->./cminus\_lex <input filename>

* Code explanation

1. util.c: void printToken(): C-MINUS의 spec에 맞게 token type과 그에 맞는 print할 string을 설정해준다.
2. scan.c: TokenType getToken(): DFA를 통해 어떤 Tokentype인지 결정한다.
   1. start state의 input으로 올 수 있는 character을 통해 다음 state를 판단한다.
      1. 숫자가 들어오면 숫자가 되는 과정인 INNUM state 로 transition 된다
      2. 알파벳(대소문자)가 들어오면 ID token으로 판단하고 그 과정인 INID로 transition 된다.
      3. 공백이 들어오면 무시한다.
      4. ‘!’ 가 들어오면 NE token이 될 가능성이 있다고 판단하고 INNE state로 transition 된다
      5. ‘<’가 들어오면 LT token 혹은 LE token 중 하나 가 될 가능성이 있으므로 INLT state로 transition 된다.
      6. ‘>’가 들어오면 GT token 혹은 GE token 중 하나가 될 가능성이 있으므로 INGT state 로 transition 된다.
      7. ‘/’가 들어오면 OVER token 혹은 comment 중 하나이므로 우선 주석이라 가정하고 INCOMMENT\_ state로 transition 된다. 이 과정에서 ‘/’는 저장하지 않는다.
   2. IN- state에 input으로 들어오는 character에 따라 error인지 여부 혹은 다음 state를 결정한다.
      1. INNE state는 ‘!’를 확인했으므로 ‘=’가 아니면 다른 token이 될 가능성이 없으므로 오류가 발생한다. ‘!’이 들어오면 NE token으로 판단하고 DONE state로 transition된다.
      2. INLT state는 ‘<’를 확인했다. ‘=’가 들어오면 LE token으로 판단한다. 만약 아니라면 LT Token으로 판단한다. 두 경우 모두 symbol을 결정했으므로 DONE state로 transition 된다.
      3. INGT state, INEQ는 작동방식위 INLT와 같다.
      4. INCOMMENT\_ state는 ‘/’를 확인했다. 만약 ‘\*’이 들어오면 comment의 시작이라 판단하고 해당 character을 무시하고 INCOMMENT state로 transition된다. 만약 그렇지 않다면 division symbol 이므로, 기존에 저장하지 않았던 ‘/’를 저장하고 OVER token이라고 판단 후 DONE state로 transition 된다.
      5. INCOMMENT state는 “/\*”을 통해 주석을 작성중인 state이다. 만약 ‘\*’가 들어오면 주석이 끝날 가능성이 있으므로 INCOMMNET\_\_ state로 transition 된다.
      6. INCOMMNET\_\_ state 는 ‘\*’이 들어와 있는 상태에서 주석이 끝날지의 여부를 결정한다. ‘/’가 들어오면 comment가 종료되어 START state로 transition 된다. 그렇지 않으면 comment를 작성중인 상태이므로 다시 COMMENT state 로 transition 된다.
      7. INNUM state에서는 숫자가 아닌 character가 들어오면 DONE state로 간다음 NUM token이라고 결정한다.
      8. INID state에서는 알파벳도 아니고 숫자도 아닌 state가 들어오면 DONE state로 간 다음 ID token이라고 결정한다.
3. main.c: 과제에 적절한 flag를 설정한다.
   1. NO\_PARSE, TraceScan 을 TRUE로 설정한다.
4. globals.h: C-MINUS의 명세에 맞는 token들을 선언한다.
5. cminus.l: lexer의 input으로 들어가는 file이다.
   1. Definition section modification
      1. identifier의 regular expression으로 {letter}({letter}|{digit})\* 을 사용함으로 letter다음 letter 혹은 숫자가 0개이상 나오면 identifier로 판단하도록 하였다.
   2. Rules section modification
      1. Longest matching rule을 위해 더 긴 token을 짧은 token위에 배치하였다.
      2. 과제 명세에 맞는 Token들을 추가하였다.
      3. “/\*” token을 발견하면 “\*/”을 발견할 때까지 input을 실행하도록 코드를 변경하였다. ‘\*’을 발견하면 ready flag가 1이 되고, 그 이후 곧바로 ‘/’을 발견하면 do-while문이 종료된다. 만약 ‘\*’후 다른 input이 들어오면 ready변수를 다시 0으로 설정하는 방식으로 구현하였다.

* Example and Screenshot

input textfile: test.1.txt, test.2.txt

1. output of input file: test.1.txt

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

1. output of input file: test.2.txt

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명