

프로젝트 #1

Scanner 구현

기한:

소스코드 (4/14 수요일 밤 11시 59분)

문서(4/19 월요일 시험시간에 제출)

-
- ❑ 조교: 김동휘 석사과정(paka@kakao.com)
 - ❑ 프로그램 1차 채점은 조교가 할 예정이고, 수행환경이 Visual Studio2019입니다.
 - ❑ 혹시 수행환경 관련하여 문제가 있는 학생은 조교에게 이메일로 연락하여 조율하세요.

프로젝트 #1 Scanner 구현

□ C- language

- (교재) Appendix A , A.1 Lexical Convention of C- 참조
- 구현환경: Visual Studio2019에서 채점할 예정.
- 입력
 - sample 프로그램 2개(교재 496-497페이지) – LMS게시판에 업로드할 예정
 - 그외 에러 토큰을 포함한 임의의 C- program으로 테스트함.
- 출력
 - 출력양식은 교재(원서) 79페이지의 그림 2.12(다음 페이지 참조)
 - 지정한 이름의 text파일: (지정한 이름의 의미는 ‘테스트’부분 참조)
 - **에러토큰의 경우, 해당 라인 번호와 "Error: 해당 error string"으로 출력하면 됩니다. 결과 끝에 나타낼 필요 없고 스캐닝 하는 도중에 다른 토큰처럼 출력함.)

Figure 2.12

Output of scanner given the
TINY program of Figure 2.11
as input

```
TINY COMPILATION: sample.tny
1: { Sample program
2:   in TINY language -
3:   computes factorial
4: }
5: read x; { input an integer }
   5: reserved word: read
   5: ID, name= x
   5: ;
6: if 0 < x then { don't compute if x <= 0 }
   6: reserved word: if
   6: NUM, val= 0
   6: <
   6: ID, name= x
   6: reserved word: then
7:   fact := 1;
   7: ID, name= fact
   7: :=
   7: NUM, val= 1
   7: ;
8:   repeat
   8: reserved word: repeat
```

```

9:      fact := fact * x;
9: ID, name= fact
9: :=
9: ID, name= fact
9: *
9: ID, name= x
9: ;
10:     x := x - 1
10: ID, name= x
10: :=
10: ID, name= x
10: -
10: NUM, val= 1
11:  until x = 0;
11: reserved word: until
11: ID, name= x
11: =
11: NUM, val= 0
11: ;
12:  write fact ( output factorial of x )
12: reserved word: write
12: ID, name= fact
13: end
13: reserved word: end
14: EOF

```

– 제출물:

- (1) 파일: 소스파일, 실행파일을 학번이름.zip으로 압축하여 LMS과제게시판에 업로드. (예:20210000홍길동.zip)
 - 소스파일의 이름은 scan.c(cpp), 실행파일의 이름은 scan.exe
- (2) 문서:
 - DFA : DFA 및 다음에 대해 Scanning하는 과정을 씀
 - (예제 문장) `if (score! =200) /* this includes error */ num=98F;`
 - 즉, inputbuffer에서 한 character씩 읽어와서 어떤 상태로 가고, 어떤 토큰을 출력하는지 자세히 쓸 것.
 - 구현에 대한 설명:
 - table driven 방식으로 한 경우:
 - table과 소스코드 첨부(적절한 코멘트 추가)
 - 교재 두번째(better method)로 한 경우:
 - getToken함수(scan.c 참고)에 적절한 코멘트를 추가하여 첨부
 - sample 프로그램 2개(교재 496-497페이지: 1.c, 2.c)를 돌렸을 때의 실행결과 첨부

– 기한:

- (파일) 4월 14일 (수) 자정까지 LMS과제게시판에 업로드
 - delayed → 감점(하루 당 -20%)
 - 소스 파일만 압축해서 제출 (프로젝트 파일을 전체로 제출할 필요 없음)
- (문서) 4월 19일 (월) 시험일, 제출
- `c:>filename inputFile outputFile: input` 파일에 적용한 결과가 `output` 파일로 나오도록
- (예) `scan a.c a.txt`
→ a.c를 입력으로 받아 스캔한 결과를 a.txt에 씀.

채점기준

- 보고서:
 - 빠뜨린 항목이 있으면 감점,
 - 내용 충실도에 따라 가,감점
- 프로그램:
 - 컴파일이 안되면 0점,
 - 타인의 코드를 카피한 경우, 원본 및 카피본 모두 0점 처리함.
 - 복제 체크 프로그램을 돌려봄.
 - 두 샘플 프로그램에 대해 실행한 결과가 틀리면 감점,
 - 에러토큰을 포함한 프로그램이 수행되지 않으면 감점.
- 채점은 크게 세 항목으로 구성
 - 1. 보고서
 - 정상프로그램 결과가 나오지 않으면 보고서 점수도 0점.
 - 2. 정상 프로그램 (1.c, 2.c ➔ LMS에 올림)
 - 3. 에러 프로그램(다수의 에러를 감지하고 출력하는지 체크)

Scanner구현 보충설명

프로젝트 #1 Scanner 구현

- Lexical convention of C-
 - 1. keywords: else, if, int, return, void, while
 - 2. Special symbols: + - * / < <= > >= == != = ; , () [] { } /* */
 - 3. ID= letter letter*, NUM = digit digit*
 - letter= a|..|z|A|..|Z, digit=0|..|9
 - lower and upper case letters are distinct
 - 4. **White space** consists of blanks, newlines, and tabs. White space is ignored except that it must separate ID's, NUM's, and keywords.
 - 5. **Comments** are surrounded by /* ... */. Comments can be placed anywhere white space can appear (that is, comments cannot be placed within tokens) and may include than one line. Comments may not be nested.

1. keywords: else, if, int, return, void, while
2. Special symbols: + - * / < <= > >= == != = ; , () [] { } /* */

3. ID= letter letter*, NUM = digit digit*

- letter= a|..|z|A|..|Z, digit=0|..|9
- lower and upper case letters are distinct

- (C-언어의 공백처리 지침 4번) White space consists of blanks, newlines, and tabs. White space is ignored except that it must separate **ID's**, **NUM's**, and **keywords** (추가로 **special symbol**).

– 공백 무시(없다고 생각)

- x + y → x+y
- xyz = 3456 → xyz=3456

– 공백 무시하지 않음

- abc xyz → "abc" "xyz" (ID ID)
- if return → "if" "return" (keyword keyword)
- abc 1234 → "abc" "1234" (ID NUM)
- 561 1234 → "561" "1234" (NUM NUM)
- > = → ">" "=" (GT ASSIGN)

1. keywords: else, if, int, return, void, while

2. Special symbols: + - * / < <= > >= == != = ; , () []

{ } /* */

3. ID= letter letter*, NUM = digit digit*

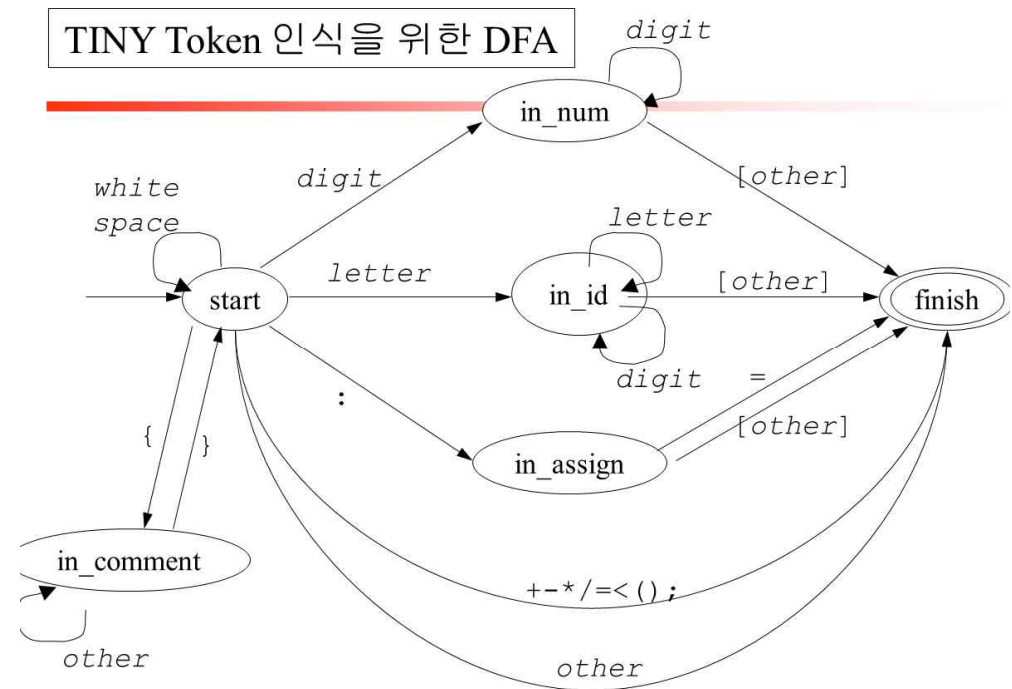
- letter= a|..|z|A|..|Z, digit=0|..|9
- lower and upper case letters are distinct

- 5. Comments are surrounded by /* ... */. Comments can be placed anywhere white space can appear (that is, comments cannot be placed within tokens) and may include than one line. Comments may not be nested.

- xyz!=10 /* comment.... */ → OK
- xyz!/* comment.... */=10 → X!
 - xyz(ID) !(ERROR) =(ASSIGN) 10(NUM)

number 다음 ID가 오는 경우

- 111 aaa (즉 111과 aaa사이에 공백) --> 111(NUM), aaa (ID)
- 111#aaa --> 111(NUM) #(ERROR) aaa(ID)
- (TINY 의 경우) 111aaa --> 111(NUM), aaa(ID)
 - 111까지 INNUM상태에 있다가,
 - lookahead로 a를 만나면 DONE상태로 가서
 - 111을 NUMBER로 처리합니다.
 - 이후 다시 aaa의 a부터 처리
- (C-의 경우)에러토큰으로 인식
 - 111aaa → 111aaa(ERROR)



주석 처리 관련(1)

- 제대로 된 주석은 컴파일러가 무시를 하게 됩니다.
 - 샘플파일에서 `/* ... */` 부분은, 교재의 결과예시와 같이 그대로 출력하게 되며 (토큰화 할 필요 없이)
 - 나중에 파서는 코멘트 부분은 따로 파싱하지 않음. TINY언어와 마찬가지로 코멘트 부분은 무시하면 됨

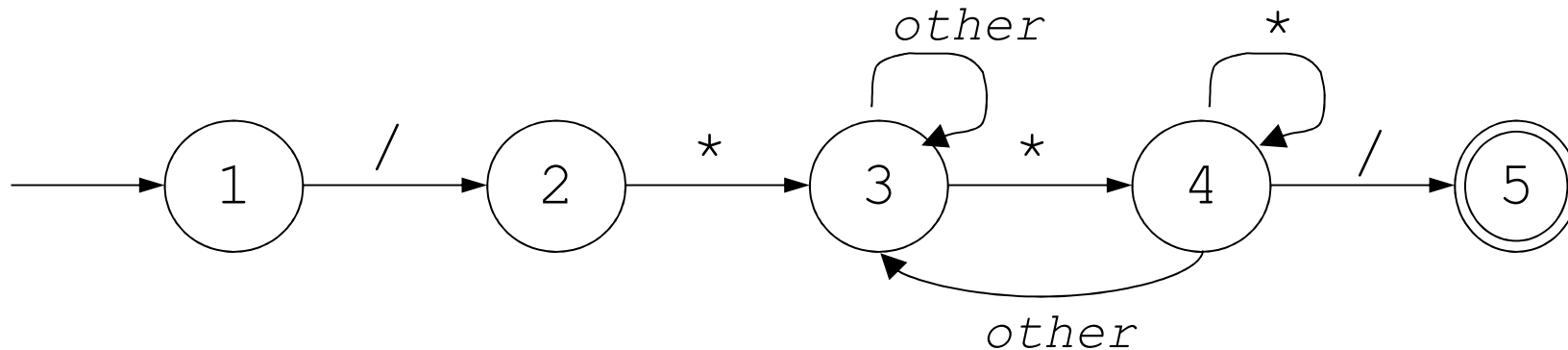
주석 처리 관련(2)

□ (예) `/* abc * /`

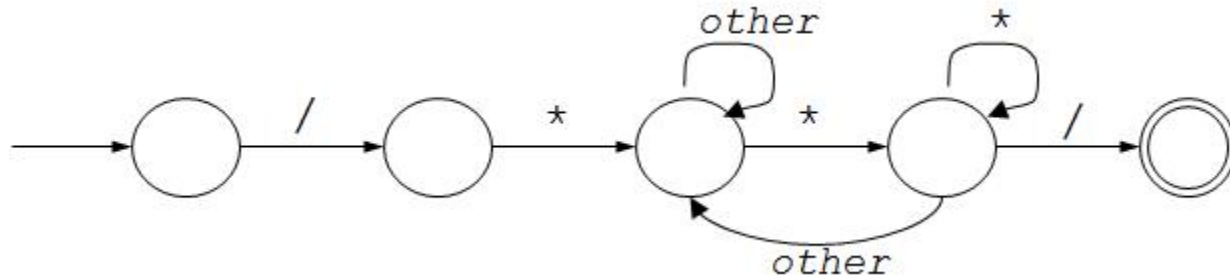
– `'* /'` 부분이 `*/` 로 인식해야 하는가

– 아래 dfa 참조➔

- `"/* abc *`까지 수행하면 4번 상태로 감
- 4번에서 공백을 만나면 `other`에 의해 3번으로 감.
- 결국 3번 상태에 계속 머무르게 됨.



주석처리 관련 (3)



- /* */의 경우
 - 주석 처리 시작 dfa로 갔다가
 - 종료되지 못하고 끝나게 됨 (scan할 대상은 없으나 finish상태가 안됨)
 - ➔ 오류 메시지 출력: "stop before ending"
- / * */의 경우
 - "/", "*", "*", "/"로 하면 됨.

주석 처리 관련(4)

- special symbol에서 두 개의 캐릭터로 이루어진 경우,
즉 \leq , \geq , $=$, $!=$ 들은 두 개가 연속하여 붙은 것이 하나의 토큰임.
 - (예) " $< \quad =$ "
 - 공백 무시하여 " \leq " (X)
 - " $<$ " 와 " $=$ " ? (O)
 - 즉, 스캐너는 읽어나가면서 " $<$ "토큰을 읽고 공백은 버리고 " $=$ "토큰을 읽는 것임.
 - (예) " $a \quad \leq b$ "
 - " $a \leq b$ " 로 보라는 뜻임.

예약어 처리 및 **symbol table**

- ❑ **symbol table**은 지금 구축하지 않아도 됨.
- ❑ 예약어의 경우,
 - 예약어 테이블을 만든 다음
 - 일단 **ID**로 인식 후
 - 예약어 테이블에서 찾아본다.