컴파일러

월/목 B

과제#1

학과: 소프트웨어
학번:201520980
이름: 정진선
제출일: 2019년 4월 1일

MINI-C Lexical Analyzer

1. 서론

1.1 개요

- Flex를 이용하여 MINI-C token을 인식하는 Lexical Analyzer 구현
- Token 별 Transition Diagram 및 Regular Expression 표현

구현된 부분

- Keyword, Identifer, Integer, Double, String, Operation, Symbols 별 토큰 모두 처리
- Integer 하위 10자리, ID 상위 16자 처리
- Token List 출력
- 연결리스트를 활용한 Symbol Table 및 Symbol Table 구현 및 출력

구현되지 않은 부분

- Int, Double 의 범위 처리

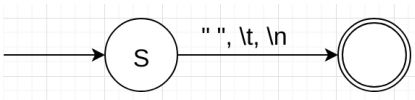
2. 문제 분석

2.1 token의 종류

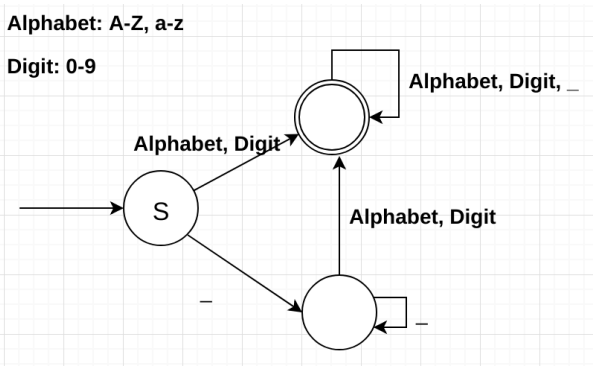
BLANK [“ “, \n,\t]
IDENTIFIER [Keyword, Identifier]
– Keyword: double, int, str, if, while, return, print
- TOKEN : DOUBLE, INT, STR, IF, WHILE, RETURN, PRINT
INTEGER (하위 10자리 정수값)
DOUBLE (실수값)
STRING [“*”]
OPERATOR [+,-,*,/,=,<,<=,>,>=,==,!=]
SYMBOL [“, \, (,), ; {, }]
COMMENT [//, /**/]
EXCEPTION [기타]

2.2 transition diagram

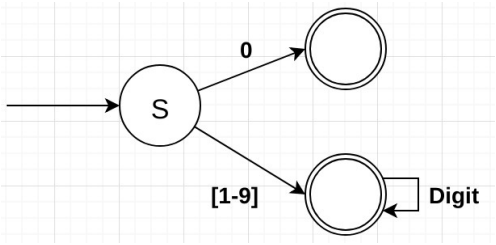
BLANK



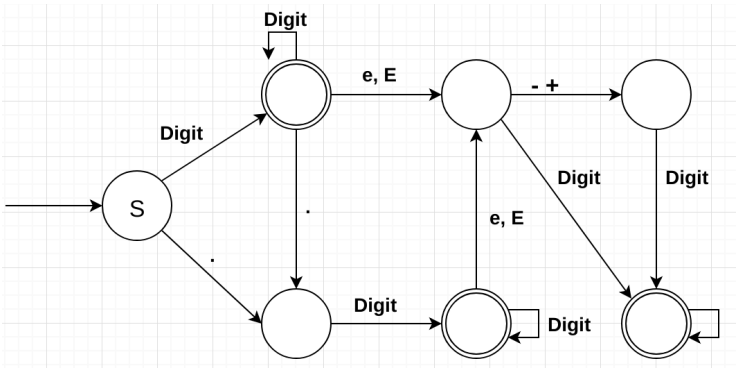
IDENTIFIER



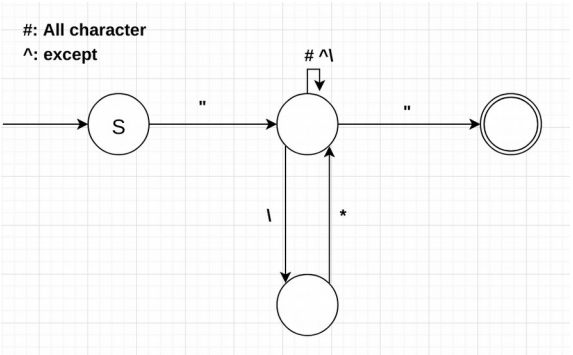
INTEGER



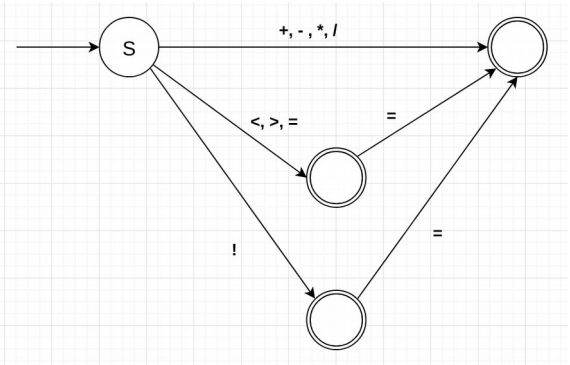
DOUBLE



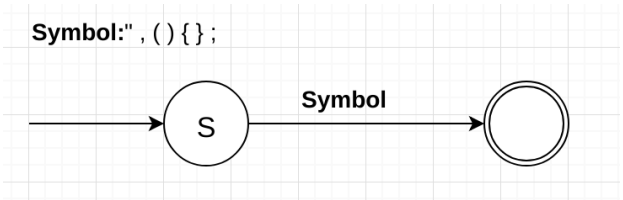
STRING



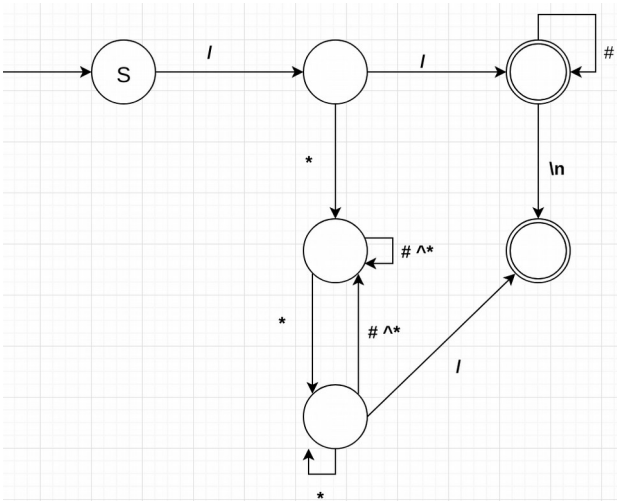
OPERATOR



SYMBOL



COMMENTS



2.3 regular expression

BLANK	" "+|[\t\n]+
IDENTIFIER	([a-zA-Z]|(_+[0-9A-Za-z]))[_A-Za-z0-9]*
INTEGER	0|([1-9][0-9]*)
DOUBLE	(([0-9]+)|([0-9]*\.[0-9]+))([Ee][-+]?[0-9]+)?
STRING	\"([^\"\\]|\\.)*\"
OPERATOR	([\+\-*\/])|(\!\=)|([<>\=]\=?)
SYMBOL	\"|\,|\(|\)|\{|\}|\;
COMMENT	\/(*([^*]|((*)+[^*\/]))*(*)+\/|(\/[^\n]*\n))
EXCEPTION	.|\n

3. 설계

3.1 중요 자료구조 : 연결리스트

1. Linked List → Symbol Table, String Table의 각 Symbol들을 Linked List로 유지함

```
struct symbol{
	char name[ID_LENGTH]; //변수의 이름 저장
	void* address; //주소값 저장 (변수의 실제 값을 저장하고 있는 주소, 아직 사용 X)
	int type; //어떤 Type인지 저장 (아직 사용X)
	struct symbol* next; // 다음 Symbol을 나타내는 Pointer 값
};

struct string_symbol{
	char name[STRING_LEGNTH]; //STRING의 길이
	void* address; //(아직 사용X)
	struct string_symbol* next; //다음 Symbol을 나타내는 Pointer 값
};
struct symbol_table{
	int size; //Symbol Table의 Symbol의 개수
	struct symbol* first_symbol; // 처음 Symbol을 나타내는 Pointer값
};
struct string_table{
	int size; //String Table의 Symbol의 개수
	struct string_symbol* first_symbol; // 처음 Symbol을 나타내는 Pointer 값
};
```

- Keyword는 우선적으로 IDENTIFER로 구분한 뒤, keyword에 포함되어 있는지 확인하는 구조로 설계했습니다.

4. 입력 데이터

4.1 ex1.txt (정상)

int double str if while return print
identifer a _ab a12345678901234567
1 100 12345678901234567890 100 1 12345678901234567890
1.123123 1.23e+123 1.23e-123 1.23e123
"STRING1" "STRING2" "STRING3" "STRING1" "STRING2" "STRING3"
+ - * / = > >= < <= == != " , ( ) ; ( )
//comment1
/*comment2*/
/*comment3
abcddd*/

[ID, String들은 Table에 모두 저장되었는지 확인하기 위해 일부 중복된 데이터를 {1,100,…
STRING1,STRING2,STING3} 추가했습니다]

4.2 ex2.txt (비정상)

! # $ % ^ & [ ] @ ~ ` ? :
_____a

idlengthmorethan16 12345678901234567890

[인식되지 않는 토큰들은 모두 <ERROR> LEXEME 형태로 표현했습니다]

5. 결과 데이터

5.1 ex1.txt

5.1.1TOKEN LIST

```
TOKEN                                   LEXEME
<INT, >                                 int
<DOUBLE, >                                      double
<STR, >                                 str
<IF, >                                  if
<WHILE, >                                       while
<RETURN, >                                      return
<PRINT, >                                       print
<ID,1>                                  identifer
<ID,2>                                  a
<ID,3>                                  _ab
<ID,4>                                  a123456789012345
<INTEGER,1>                                     1
<INTEGER,100>                                   100
<INTEGER,1234567890>                                    1234567890
<INTEGER,100>                                   100
<INTEGER,1>                                     1
<INTEGER,1234567890>                                    1234567890
<DOUBLE,1.123123>                                       1.123123
<DOUBLE,1.23e+123>                                      1.23e+123
<DOUBLE,1.23e-123>                                      1.23e-123
<DOUBLE,1.23e123>                                       1.23e123
<STRING,1>                                      "STRING1"
<STRING,2>                                      "STRING2"
<STRING,3>                                      "STRING3"
<STRING,1>                                      "STRING1"
<STRING,2>                                      "STRING2"
<STRING,3>                                      "STRING3"
<PLUS, >                                        +
<MINUS, >                                       -
<MULTI, >                                       *
```

```
<DIV, >                         /
<EQ, >                          =
<GT, >                          >
<GE, >                          >=
<LT, >                          <
<LE, >                          <=
<IQ, >                          ==
<NQ, >                          !=
<QUOTES, >                              "
<COMMA, >                               ,
<LPAREN, >                              (
<RPAREN, >                              )
<SEMICOLON, >                           ;
<LPAREN, >                              (
<RPAREN, >                              )
<COMMENT>                               //comment1

<COMMENT>                               /*comment2*/
<COMMENT>                               /*comment3
abcddd*/
```

5.1.2 SYMBOL TABLE

INDEX	SYMBOLS
1	identifer
2	a
3	_ab
4	a1234567890123456789

5.1.3 STRING TABLE

INDEX	STRINGS
1	"STRING1"
2	"STRING2"
3	"STRING3"

5.2 ex2.txt

5.2.1 TOKEN LIST

```
TOKEN                                   LEXEME
<ERROR>                                 !
<ERROR>                                 #
<ERROR>                                 $
<ERROR>                                 %
<ERROR>                                 ^
<ERROR>                                 &
<ERROR>                                 [
<ERROR>                                 ]
<ERROR>                                 @
<ERROR>                                 ~
<ERROR>                                 `
<ERROR>                                 ?
<ERROR>                                 :
<ID,1>                                  _____________________
<ID,2>                                  idlengthmorethan
<INTEGER,1234567890>                                      1234567890
```

5.2.2 SYMBOL TABLE

```
INDEX     SYMBOLS
1         _____________________
2         idlengthmorethan
```

5.2.3 STRING TABLE

```
INDEX     STRINGS
```

6. 첨부자료 (별도 첨부 : lexical_analyzer.l)