형식언어

- Assignment #9 -

동국대학교 컴퓨터공학과

2016112122 이한빈

■ MiniC Scanner 확장

1. 추가 인식 키워드: char, double, string, for, switch, case, default, continue, break

2. 추가 인식 리터럴: 문자, 문자열, 실수

3. 주석: line comment (//), multiline comment (/\* \*/), document comment(/\*\* \*/)

4. 추가 연산자: ‘:’

|  |
| --- |
| enum tsymbol {  .  .  .  twhile, tlbrace, tor, trbrace, tchar, tdouble,  tstring, tfor, tswitch, tcase, tdefault, tcontinue,  tbreak, tcolon, tschar, tsstring, trealnumber, tlinecomment,  tdocumentcomment, tmultilinecomment }; |

각 Token에 고유한 ID를 부여하는 tsymbol에 keyword를 추가하였다.

|  |
| --- |
| struct tokenType {  int number; // token number  union {  char id[ID\_LENGTH];  int num;  double real;  char c;  char s[MAX\_LENGTH];  char comment[MAX\_COMMENT\_LENGTH];  } value;  }; |

Token의 number와 value를 저장하는 struct tokenType에 double real, char c, char s[], char comment 추가. real은 실수, c는 문자, s는 문자열, comment는 주석을 저장한다.

|  |  |
| --- | --- |
| double getRealNumber(int num); | 소수점 아래 숫자를 정수로 입력 받아 원래의 실수부로 반환 |
| char getCharacter(FILE\* sourceFile, char ch); | file에서 char type 문자를 하나 추출하여 반환. char type은 ‘로 싸인 형태이다. |
| char\* getString(FILE\* sourceFile, char ch); | file에서 string type 문자열을 하나 추출하여 반환. string type 은 “로 싸인 형태이다. |
| char\* getLineComment(FILE\* sourceFile, char ch); | file의 line comment를 반환 line comment는 // 뒤에 나타남. token.value로 //뒤에 문자열 저장 |
| char\* getDocComment(FILE\* sourceFile, char ch); | file의 multiline comment를 반환. multiline comment는 /\* \*/ 형식을 갖는다. token.value에는 /\* \*/ 사이의 문자열 저장.  닫는 문자 \*/이 제대로 오지 않으면 Error를 발생시켰다. |
| double getFloatingPoint(FILE\* sourceFile, char ch, int num); | file에서 부동소수 반환. e가 나오면 호출되는 함수. |
| void deleteStar(char\* s, char\* s2, int maxLen); | multiline comment를 token.value에 저장할 때 /\* \*/ 를 앞뒤 같은 개수로 연속된 star삭제 |

|  |
| --- |
| case '/':  ch = fgetc(sourceFile);  if (ch == '\*') // text comment  {  char temp[MAX\_COMMENT\_LENGTH];  strcpy\_s(temp, MAX\_COMMENT\_LENGTH, getDocComment(sourceFile, ch));  if (\*temp == '\*') // document comment  {  \*temp = \*(temp + 1);  token.number = tdocumentcomment;  strcpy\_s(token.value.comment, MAX\_COMMENT\_LENGTH, temp);  }  else // multilinecomment  {  token.number = tmultilinecomment;  strcpy\_s(token.value.comment, MAX\_COMMENT\_LENGTH, temp);  }  } |
| multiline comment와 document comment는 공통된 과정을 거친다. getDocComment를 통해 반환된 문자열이 \*로 시작하면 token을 document comment로 인식하고, 그렇지 않다면 multiline comment로 인식했다. document comment의 경우 앞의 \*를 제거해주기 위해 \*temp = \*(temp + 1); 왼쪽과 같은 과정을 거쳤다.  **실제 스캐너에서는 주석의 내용까지는 인식하지 않는다. 그러나, 이번 과제에서는 주석의 내용을 보여주기 위해 token.value.comment에 주석 내용을 저장하고 출력했다.** |

■ Result

|  |
| --- |
| test2.mc |
| /\*\*  document comment  \*/  /\* multiline comment\*/  int main()  {  double d1 = 1.23;  double d2 = 45.623e-4;  double d3 = 45.632e+5;  double d4 = 123e-5;  char c = 'a'; // comment  string str = "abc"; // comment  int i = 0;  switch(i)  {  case 0:  continue;  default:  break;  }  for (int i = 0; i < 10; i++)  break;  } |
| test2.out |
| Token docstringcomment ( 54, \n document comment\n)  Token multilinecomment ( 55, multiline commentmment\n)  Token int ( 33, 0 )  Token %ident ( 4, main )  Token ( ( 7, 0 )  Token ) ( 8, 0 )  Token { ( 37, 0 )  Token double ( 41, 0 )  Token %ident ( 4, d1 )  Token = ( 23, 0 )  Token %real\_number ( 52, 1.230000)  Token double ( 41, 0 )  Token %ident ( 4, d2 )  Token = ( 23, 0 )  Token %real\_number ( 52, 0.004562)  Token double ( 41, 0 )  Token %ident ( 4, d3 )  Token = ( 23, 0 )  Token %real\_number ( 52, 4563200.000000)  Token double ( 41, 0 )  Token %ident ( 4, d4 )  Token = ( 23, 0 )  Token %real\_number ( 52, 0.001230)  Token char ( 40, 0 )  Token %ident ( 4, c )  Token = ( 23, 0 )  Token %char ( 50, a)  Token ; ( 20, 0 )  Token linecomment ( 53, comment)  Token string ( 42, 0 )  Token %ident ( 4, str )  Token = ( 23, 0 )  Token %string ( 51, abc)  Token linecomment ( 53, comment)  Token int ( 33, 0 )  Token %ident ( 4, i )  Token = ( 23, 0 )  Token %number ( 5, 0 )  Token switch ( 44, 0 )  Token ( ( 7, 0 )  Token %ident ( 4, i )  Token ) ( 8, 0 )  Token { ( 37, 0 )  Token case ( 45, 0 )  Token %number ( 5, 0 )  Token continue ( 47, 0 )  Token ; ( 20, 0 )  Token default ( 46, 0 )  Token : ( 49, 0 )  Token break ( 48, 0 )  Token ; ( 20, 0 )  Token } ( 39, 0 )  Token for ( 43, 0 )  Token ( ( 7, 0 )  Token int ( 33, 0 )  Token %ident ( 4, i )  Token = ( 23, 0 )  Token %number ( 5, 0 )  Token %ident ( 4, i )  Token < ( 21, 0 )  Token %number ( 5, 10 )  Token %ident ( 4, i )  Token ++ ( 12, 0 )  Token ) ( 8, 0 )  Token break ( 48, 0 )  Token ; ( 20, 0 )  Token } ( 39, 0 )  Token eof ( 29, 0 ) |

위는 임의로 만든 test2.mc를 어휘분석한 결과이다. 추가된 literal과 keyword에 대하여 어휘분석이 정상적으로 된 것을 확인할 수 있다.