C언어를 이용한 어휘분석기 구현

소프트웨어공학과 201513820 김민규

**C 파일**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <ctype.h>

#define NO\_KEWORDS 7

#define ID\_LENGTH 12

enum tsymbol

{

   tnull = -1,

   tnot,   tnotequ,   tmod,   tmodAssign,   tident,

   tnumber,   tand,   tlparen,   trparen,   tmul,

   tmulAssign,   tplus,   tinc,   taddAssign,   tcomma,

   tminus,   tdec,   tsubAssign,   tdiv,   tdivAssign,

   tsemicolon,   tless,   tlesse,   tassign,   tequal,

   tgreat,   tgreate,   tlbracket,   trbracket,   teof,

   tconst,   telse,   tif,   tint,   treturn,

   tvoid,   twhile,   tlbrace,   tor,   trbrace

};

struct tokenType

{

   int number;

   char sym[2];

   union

   {

      char id[ID\_LENGTH];

      int num;

   } value;

} a;

char \*keyword[NO\_KEWORDS] = {

   "const", "else", "if", "int", "return", "void", "while"

};

enum tsymbol tnum[NO\_KEWORDS] = {

   tconst, telse, tif, tint, treturn, tvoid, twhile};

void lexicalError(int n)

{

   printf(" \*\*\*   Lexical Error : ");

   switch (n)

   {

   case 1:

      printf("an identifier length must be less than 12. \n");

      break;

   case 2:

      printf("next character must be &.\n");

      break;

   case 3:

      printf("next character must be |. \n");

      break;

   case 4:

      printf("invalid character!!! \n");

      break;

   }

}

int superLetter(char ch)

{

   if (isalpha(ch) || ch == '\_')

      return 1;

   else

      return 0;

}

int superLetterOrDigit(char ch)

{

   if (isalnum(ch) || ch == '\_')

      return 1;

   else

      return 0;

}

int hexValue (char ch) {

   switch (ch)

   {

   case '0' : case '1' : case '2' : case '3' : case '4' :

   case '5' : case '6' : case '7' : case '8' : case '9' :

      return (ch - '0');

   case 'A' : case 'B' : case 'C' : case 'D' : case 'E' : case 'F' :

      return (ch - 'A' + 10);

   case 'a' : case 'b' : case 'c' : case 'd' : case 'e' : case 'f' :

      return (ch - 'a' + 10);

   default : return -1;

   }

}

int getIntNum(char firstCharacter)

{

   int num = 0;

   int value;

   char ch;

   if (firstCharacter != '0')

   {

      ch = firstCharacter;

      do

      {

         num = 10 \* num + (int)(ch - '0');

         ch = getchar();

      } while (isdigit(ch));

   }

   else

   {

      ch = getchar();

      if ((ch >= '0') && (ch <= '7'))

         do {

            num = 8\*num + (int)(ch - '0');

            ch = getchar();

         } while ((ch >= '0') && (ch - '7'));

      else if ((ch == 'X') || (ch == 'x')) {

         while ((value = hexValue(ch = getchar())) != -1)

            num = 16\*num + value;

      }

      else num = 0;

   }

   ungetc(ch, stdin);

   return num;

}

struct tokenType scanner()

{

   struct tokenType token;

   int i, index;

   char ch, id[ID\_LENGTH];

   token.number = tnull;

   do

   {

      while (isspace(ch = getchar()))

         ;

      if (superLetter(ch))

      {

         i = 0;

         do

         {

            if (i < ID\_LENGTH)

               id[i++] = ch;

            ch = getchar();

         } while (superLetterOrDigit(ch));

         if (i >= ID\_LENGTH)

            lexicalError(1);

         id[i] = '\0';

         ungetc(ch, stdin);

         for (index = 0; index < NO\_KEWORDS; index++)

         {

            if (!strcmp(id, keyword[index]))

               break;

         }

         if (index < NO\_KEWORDS)

         {

            token.number = tnum[index];

         }

         else

         {

            token.number = tident;

            strcpy(token.value.id, id);

         }

      }

      else if (isdigit(ch))

      {

         token.number = tnumber;

         token.value.num = getIntNum(ch);

      }

      else

         switch (ch)

         {

         case '/':

            ch = getchar();

            if (ch == '\*')

               do

               {

                  while (ch != '\*')

                     ch = getchar();

                  ch = getchar();

               } while (ch != '/');

            else if (ch == '/')

               while (getchar() != '\n')

                  ;

            else if (ch == '=')

               token.number = tdivAssign;

            else

            {

               token.number = tdiv;

               ungetc(ch, stdin);

            }

            break;

         case '!':

            ch = getchar();

            if (ch == '=')

               token.number = tnotequ;

            else

            {

               token.number = tnot;

               ungetc(ch, stdin);

            }

            break;

         case '%':

            ch = getchar();

            if (ch == '=')

               token.number = tmodAssign;

            else

            {

               token.number = tmod;

               ungetc(ch, stdin);

            }

            break;

         case '&':

            ch = getchar();

            if (ch == '&')

               token.number = tand;

            else

            {

               lexicalError(2);

               ungetc(ch, stdin);

            }

            break;

         case '\*':

            ch = getchar();

            if (ch == '=')

               token.number = tmulAssign;

            else

            {

               token.number = tmul;

               ungetc(ch, stdin);

            }

            break;

         case '+':

            ch = getchar();

            if (ch == '+')

               token.number = tinc;

            else if (ch == '+')

               token.number = taddAssign;

            else

            {

               token.number = tplus;

               ungetc(ch, stdin);

            }

            break;

         case '-':

            ch = getchar();

            if (ch == '-')

               token.number = tdec;

            else if (ch == '=')

               token.number = tsubAssign;

            else

            {

               token.number = tminus;

               ungetc(ch, stdin);

            }

            break;

         case '<':

            ch = getchar();

            if (ch == '=')

               token.number = tlesse;

            else

            {

               token.number = tless;

               ungetc(ch, stdin);

            }

            break;

         case '=':

            ch = getchar();

            if (ch == '=')

               token.number = tequal;

            else

            {

               token.number = tassign;

               ungetc(ch, stdin);

            }

            break;

         case '>':

            ch = getchar();

            if (ch == '=')

               token.number = tgreate;

            else

            {

               token.number = tgreat;

               ungetc(ch, stdin);

            }

            break;

         case '|':

            ch = getchar();

            if (ch == '|')

               token.number = tor;

            else

            {

               lexicalError(3);

               ungetc(ch, stdin);

            }

            break;

         case '(':

            token.number = tlparen;

            break;

         case ')':

            token.number = trparen;

            break;

         case ',':

            token.number = tcomma;

            break;

         case ';':

            token.number = tsemicolon;

            break;

         case '[':

            token.number = tlbracket;

            break;

         case ']':

            token.number = trbracket;

            break;

         case '{':

            token.number = tlbrace;

            break;

         case '}':

            token.number = trbrace;

            break;

         case 'EOF':

            token.number = teof;

            break;

         default:

            printf("Current character : %c", ch);

            lexicalError(4);

            break;

         }

   } while (token.number == tnull);

   return token;

}

void main()

{

   printf("Input String : \n");

   while (1)

   {

      a = scanner();

      if (a.number < 21)

         printf("Symbol : %s\n", a.sym);

      else if (a.number > 22 && a.number < 29)

         printf("Keyword : %s\n", a.value.id);

      else if (a.number == 21)

         printf("Number : %d \n", a.value.num);

      else if (a.number == 22)

         printf("Variable : %s \n", a.value.id);

      else

      {

      }

   }

}

**실행 결과**

