UCS 1602 - Compiler Design Assignment – 1- Lexical Analyser Using C

**Swetha Saseendran**

**CSE-C**

**185001183**

**Aim:**

To write a program to implement Lexical Analyser and stimulate its functionalities in C.

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

char\* parse(char \*str);

void classify(char \*str);

int isKeyword(char \*token);

int isIdentifier(char \*token);

int isConstant(char \*token);

int isRelational(char \*token);

int isLogial(char \*token);

int isSpecialChar(char c);

int isPPDIR(char \*token);

int isAOP(char \*token,int id, int len);

int main(void)

{

    char \*line = (char\*)malloc(sizeof(char)\*64);

    FILE \*fptr;

    char str[512], file[64];

    fptr = fopen("in.txt", "r");

    while (fgets(str, sizeof(str), fptr))

    {

        line = parse(str);

        classify(line);

    }

    return 0;

}

char\* parse(char \*str)

{

    int i, len = strlen(str), j = 0;

    char \*str1 = (char\*)malloc(sizeof(char)\*64);

    int flag = 0;

    for (i = 0; i < len; ++i)

    {

        if((str[i] == '+' && str[i + 1] == '+') || (str[i] == '-' && str[i + 1] == '-') || str[i] == '.')

            continue;

        if ((flag == 0 && str[i] == '"'))

        {

            if (j > 0 && str1[j - 1] != ':')

            {

                str1[j++] = ':';

            }

            str1[j++] = str[i];

            flag = 1;

            continue;

        }

        if (flag)

        {

            str1[j++] = str[i];

            if (str[i] == '"')

            {

                str1[j++] = ':';

                flag = 0;

            }

            continue;

        }

        if (isspace(str[i]))

        {

            str1[j++] = ':';

        }

        else if (!isalnum(str[i]))

        {

            if (j > 0 && str1[j - 1] != ':')

            {

                str1[j++] = ':';

            }

            str1[j++] = str[i];

            str1[j++] = ':';

        }

        else

            str1[j++] = str[i];

    }

    return str1;

}

void classify(char \*str)

{

    char \*token = (char\*)malloc(sizeof(char) \* 64);

    char \*line = str;

    int c, id = 0;

    char \*classifiedAs = (char\*)malloc(sizeof(char) \* 64);

    strcat(classifiedAs, "");

    int ppdir\_flag = 0;

    int index = 0;

    while ((token = strtok\_r(line, ":", &line)))

    {

        index++;

        int len = strlen(token);

        if(isPPDIR(token))

        {

          if(ppdir\_flag == 0)

          {

            strcat(classifiedAs, "PPDIR");

            ppdir\_flag = 1;

          }

          continue;

        }

        else if (ppdir\_flag == 1 && (index == 2 || index == 3 || index == 4 || index == 5))

        {

            continue;

        }

        else if (isKeyword(token))

        {

            strcat(classifiedAs, "KW ");

        }

        else if (isIdentifier(token))

        {

            id = 1;

            strcat(classifiedAs, "ID ");

        }

        else if (strcmp(token,"=") == 0)

        {

            strcat(classifiedAs, "ASSIGN ");

        }

        else if ((c = isConstant(token)))

        {

            if (c == 1)

                strcat(classifiedAs, "NUMCONST ");

            else if (c == 2)

                strcat(classifiedAs, "CHARCONST ");

            else if (c == 3)

                strcat(classifiedAs, "STRCONST ");

        }

        else if ((c = isRelational(token)))

        {

            if (c == 1)

                strcat(classifiedAs, "EQ ");

            else if (c == 2)

                strcat(classifiedAs, "GE ");

            else if (c == 3)

                strcat(classifiedAs, "LE ");

            else if (c == 4)

                strcat(classifiedAs, "GT ");

            else if (c == 5)

                strcat(classifiedAs, "LT ");

        }

        else if (isLogical(token))

        {

            strcat(classifiedAs, "LOGOP ");

        }

        else if (isAOP(token,id,len))

        {

            strcat(classifiedAs, "AOP ");

        }

        else if (id == 1 && token[0] == '(')

        {

            int len = strlen(classifiedAs);

            classifiedAs[len - 3] = 'F';

            classifiedAs[len - 2] = 'C';

            classifiedAs[len - 1] = '\0';

            strcat(classifiedAs, " SP ");

        }

        else if (isSpecialChar(token[0]))

        {

            strcat(classifiedAs, "SP ");

        }

    }

    printf("%s\n", classifiedAs);

    return;

}

int isKeyword(char \*token)

{

    int i;

    char keywords[][32] = {"int", "char", "float",

    "if", "else", "for", "do", "while","void"};

    for (i = 0; i < 32; ++i)

    {

        if (strcmp(token, keywords[i]) == 0)

            return 1;

    }

    return 0;

}

int isIdentifier(char \*token)

{

    if (!isalpha(token[0]))

      return 0;

    return 1;

}

int isConstant(char \*token)

{

    int i = 0;

    int constant = -1;

    int len = strlen(token);

    int decimal\_pt = 0;

    // NUMCONST

    for (i = 0; i < len && decimal\_pt < 2; ++i)

    {

        if (token[i] == '.')

        {

            decimal\_pt++;

            continue;

        }

        if (!isdigit(token[i]))

        {

            return 0;

        }

    }

    // STRCONST

    if (token[0] == '"' && token[len - 1] == '"')

        return 3;

    // CHARCONST

    if (token[0] == '\'' && token[len - 1] == '\'' && len == 3)

        return 2;

    return 1;

}

int isRelational(char \*token)

{

  if (strcmp(token,"==") == 0 )

    return 1;

  else if (strcmp(token,">=") == 0)

    return 2;

  else if(strcmp(token,"<=") == 0)

    return 3;

  else if(strcmp(token,">") == 0)

    return 4;

  else if(strcmp(token,"<") == 0)

    return 5;

  return 0; //not a relop

}

int isLogical(char \*token)

{

  if (token[0] == '!' || strcmp(token,"&&") == 0 || strcmp(token,"||") == 0)

    return 1;

  return 0;

}

int isSpecialChar(char c)

{

    char sp[10]={'{','[','}',']',';',':',',','(',')'};

    for(int i=0;i<9;i++)

    {

        if(sp[i]==c)

            return 1;

    }

    return 0;

}

int isPPDIR(char \*token)

{

  if (strcmp(token,"#") == 0 || strcmp(token,"include") == 0 || strcmp(token,"<:studioh:>") == 0)

    return 1;

  return 0;

}

int isAOP(char \*token,int id,int len)

{

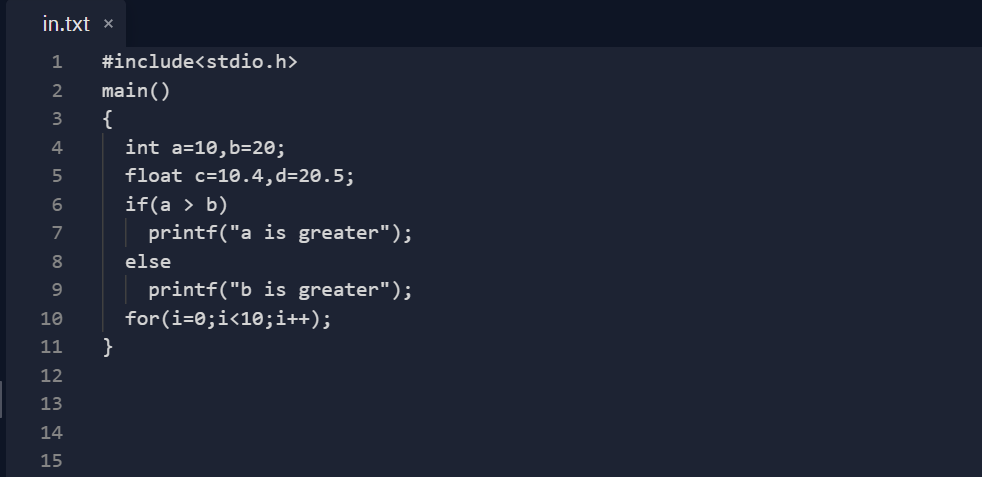
  if ((strcmp(token,"++") == 0 || strcmp(token,"--") == 0 ) || (len == 1 && (token[0] == '+' || token[0] == '-' || (id == 1 && token[0] == '\*') || token[0] == '/')))

    return 1;

  return 0;

}

**INPUT TEXT FILE TO PARSE (in.txt):**



**OUTPUT:** 