Kaustav Ghosh

Roll No. 29

Reg No. 180905188

Lab CD

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

typedef struct ste {

char lexeme[30];

int index;

char type[15];

int size;

} ste;

typedef struct token {

char token\_name[100];

unsigned int row,col;

} token;

typedef struct localTableData {

char fn\_name[30];

int size;

} localTableData;

FILE \*file;

int row = 1, col = 0;

ste table[10][100];

localTableData tableData[10];

int st\_index = 0;

int st\_num = -1;

int scope = 0;

char dbuf[15] = "";

int isKeyword (char \*word)

{

char \*keylist[30] = { "const", "int", "struct", "break", "else", "char",

"float","long","switch","case","enum","return","continue",

"for","signed","void","do","if","static","while","double","sizeof","short",

"unsigned","typedef", "true", "false", "bool", "printf", "scanf" };

for (int i = 0; i < 30; ++i)

{

if (strcmp(word, keylist[i]) == 0) {

return 1;

}

}

return 0;

}

int isSpecialSymbol (char ch)

{

return (strchr("[]{}(),;:.#", ch) != NULL);

}

int isOperator (char ch)

{

return (strchr("+-\*/%<>^=!", ch) != NULL);

}

token newToken(char name[50], int r, int c)

{

token t;

strcpy(t.token\_name, name);

t.row = r;

t.col = c;

return t;

}

void insertSt(char lexeme[100], char type[15], int size)

{

ste t;

strcpy(t.lexeme, lexeme);

strcpy(t.type, type);

t.size = size;

t.index = st\_index + 1;

for (int i = 0; i < st\_index; i++)

{

if (strcmp(table[st\_num][i].lexeme, lexeme) == 0)

{

return;

}

}

table[st\_num][st\_index++] = t;

tableData[st\_num].size += 1;

}

void printSt()

{

for (int j = 0; j <= st\_num; j++)

{

printf("\n------------------------------------\n");

printf("\nSymbol table for %s\n\n", tableData[j].fn\_name);

printf("Index\tLexeme\tType\tSize\n");

for (int i = 0; i < tableData[j].size; i++)

{

printf("%d\t%s\t%s\t%d\n", table[j][i].index, table[j][i].lexeme,

table[j][i].type, table[j][i].size);

}

}

}

token getNextToken()

{

char buf[30];

int idx = 0;

int ca = getc(file);

col++;

token t;

while(ca == '\n')

{

strcpy(dbuf, "");

ca = getc(file);

col = 1;

row += 1;

}

if (ca == EOF)

{

return newToken("EOF", row, col);

}

if (ca == ' ')

{

ca = getc(file);

col += 1;

}

if (isalpha(ca))

{

while (isalpha(ca) || isdigit(ca) || ca == '\_')

{

buf[idx++] = ca;

ca = getc(file);

col++;

}

ungetc(ca, file);

col--;

buf[idx] = '\0';

if (isKeyword(buf))

{

strcpy(dbuf, buf);

t = newToken(buf, row, col - strlen(buf) + 1);

}

else

{

int num = 1;

t = newToken("id", row, col - strlen(buf) + 1);

ca = getc(file);

if (ca == '(')

{

if (scope != 0)

{

insertSt(buf, "func", -1);

}

else

{

//printf("New function %s\n\n", buf);

st\_num++;

strcpy(tableData[st\_num].fn\_name, buf);

st\_index=0;

}

ungetc(ca, file);

return t;

}

if (ca == '[')

{

int pos = ftell(file);

int cb;

num = 0;

while ((cb = getc(file)) != ']')

{

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

}

fseek(file, pos, SEEK\_SET);

}

ungetc(ca, file);

int sz = 0;

if (strcmp(dbuf, "int") == 0 || strcmp(dbuf, "float") == 0) sz = 4;

else if (strcmp(dbuf, "char") == 0 || strcmp(dbuf, "bool") == 0) sz =1;

else if (strcmp(dbuf, "double") == 0 || strcmp(dbuf, "long") == 0) sz = 8;

insertSt(buf, dbuf, sz \* num);

}

}

else if (isdigit(ca))

{

t = newToken("num", row, col);

while (isdigit(ca) || ca == '.' || ca == 'e' || ca == 'E')

{

buf[idx++] = ca;

ca = getc(file);

col++;

}

ungetc(ca, file);

col--;

buf[idx] = '\0';

}

else if(ca == '"')

{

t = newToken("string", row, col);

do

{

ca = getc(file);

col++;

} while(ca != '"');

}

else if (isSpecialSymbol(ca))

{

if (ca == '{') scope++;

if (ca == '}') scope--;

buf[idx++] = ca;

buf[idx] = '\0';

t = newToken(buf, row, col);

}

else if (isOperator(ca))

{

buf[idx++] = ca;

ca = getc(file);col++;

if (ca == '=')

{

buf[idx++] = ca;

}

else

{

ungetc(ca, file);

col--;

}

buf[idx] = '\0';

t = newToken(buf, row, col - strlen(buf) + 1);

}

else if(ca == '&')

{

buf[idx++] = ca;

ca = getc(file);col++;

if (ca == '&')

buf[idx++] = ca;

else

{

ungetc(ca, file);

col--;

}

buf[idx] = '\0';

t = newToken(buf, row, col - strlen(buf) + 1);

}

else if(ca == '|')

{

buf[idx++] = ca;

ca = getc(file);

col++;

if (ca == '|')

buf[idx++] = ca;

else

{

ungetc(ca, file);

col--;

}

buf[idx] = '\0';

t = newToken(buf, row, col - strlen(buf) + 1);

}

return t;

}

int main()

{

FILE \*fa, \*fb;

int prev, ca, cb;

fa = fopen("input.c", "r");

if (fa == NULL)

{

printf("Cannot open file \n");

exit(0);

}

fb = fopen("output.c", "w+");

ca = getc(fa);

while (ca != EOF)

{

if(ca==' ' || ca == '\t')

{

putc(' ',fb);

while(ca==' ' || ca == '\t') ca = getc(fa);

ungetc(ca, fa);

}

else if (ca=='/')

{

cb = getc(fa);

if (cb == '/')

{

while(ca != '\n')

ca = getc(fa);

}

else if (cb == '\*')

{

do {

while(ca != '\*')

ca = getc(fa);

ca = getc(fa);

} while (ca != '/');

ca = getc(fa);

}

else

{

putc(ca,fb);

putc(cb,fb);

}

}

else if (ca == '#')

{

while (ca != '\n')

{

ca = getc(fa);

}

}

else putc(ca,fb);

ca = getc(fa);

}

fclose(fa);

fclose(fb);

file = fopen("output.c", "r");

printf("Tokens:\n\n");

while(1)

{

token t = getNextToken();

if (strcmp(t.token\_name, "EOF") == 0) break;

printf("<%s, %d, %d>\n", t.token\_name, t.row, t.col);

};

fclose(file);

printSt();

return 0;

}



