PROGRAMLAMA

DİLLERİ PROJE-1

**LEXİCAL ANALYZER**

YILMAZ GEZER - 05110000807

AHMET ERCAN ÜNAL - 05130001212

**Rapor Analizi**

Kullanıcıdan daha önceden var olan bir dosyanın ismi alınarak içinde ki kodlar satır satır okunur. Sonrasında Fortran 77 diline uygun olarak kodlar tokenlarına ayrılarak token listesi ekran da gösterilir ve aynı zamanda dosya\_ismi.lex uzantılı bir dosyaya ekran çıktıları da kaydedilir. Bu işlemler C dilinde gerçekleştirilir.

**Problemin çözüm yöntemi.**

Fotran 77 dilinin kurallarına uygun olarak öncelikli olan şeyler süzgeçten geçirilir. Bir satır okunduğunda satırdaki karakterler üzerinde işlem yapılır. Bir satırda maksimum 72 karakter vardır. O satırı string olarak algılanması için maksimum char array’in boyutu 73 olarak alınır. Bu temel bilgilerden sonra Adım adım yapılan işlemler aşağıdaki gibidir.

**Adım 1:** Boş karakterler geçilir.

**Adım 2:** Başlangıç karakteri C veya \* ise ve ondan sonra gelen karakter boşluk veya alt satır karakteri ise o satırın açıklama satırı olduğu anlaşılır, bundan dolayı direk olarak bir sonraki satır okunur ve tekrar 1. Adıma geri dönülür.

**Adım 3:** Satırda üzerinde işlem yapılırken aktif olan karakter tek tırnak işareti ise tekrar bir tek tırnak veya satır sonuna gelene kadar okunarak String Constant elde edilir.

**Adım 4:** Üsteki adımlara takılmazsa işlem yapmaya devam edilir, yani bir tür süzgeç gibi düşünebiliriz. Yukardaki adımlara takılmadıysa bundan sonra yapılacak işlem aktif satırın aktif karakterinin bir sembol olup olmadığı veya bir aritmatik operatör olup olmadığı gibi kontroller yapılır.

**Adım 5:** Bu adımda eğer yukarıda ki adımlardan birinde yine takılmadıysak ve bir karakter ile başlıyorsa bunun anlamı bir identifer veya bir keyword olduğudur. Kelime elde edildikten sonra daha önceden bir text dosyasından okunarak elde eldilen keywordlerim ile kelime karşılaştırılır. Eğer kelime keywordlerimden birisiyle eşleşiyorsa kelime bir keyworddür eşleme sağlanmadıysa kelime bir identiferdır.

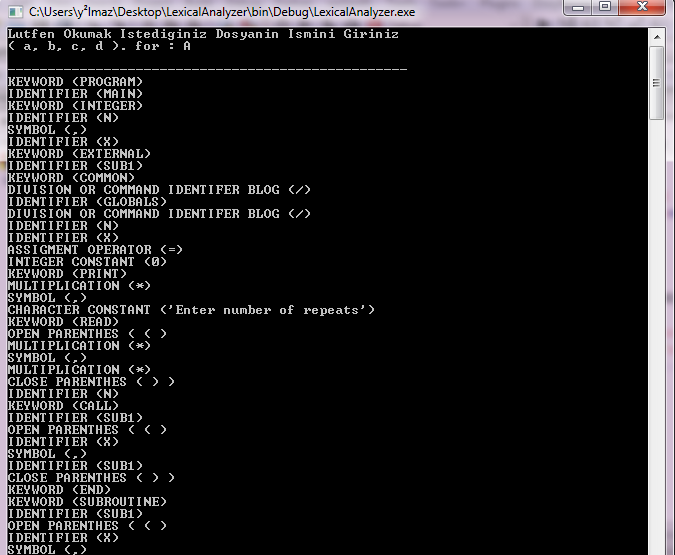
**Adım 6:** Aktif satırın aktif karakteri bir sayı ile başlıyorsa sayının sonuna kadar gidilir. Sayının içinde bir nokta varsa float constant olarak alınır yoksa integer constant olarak alınır.

**Adım 7:** Bu adımda mantıksal bir mantıksal operatörlerin kontrolleri yapılır. Mantıksal operatörler nokta ile başlayıp nokta ile biter. En az 4 en çok 6. Karakterlidir. Aktif olan karakterden itibaren en azdan en çok karaktere sahip mantıksal operatörler karşılaştırılır. Bulunduğunda bilgiler ekranda gösterilip .lex uzantılı dosyaya kaydı yapılır.

**Adım 8:** Bu adıma gelindiğinde karakterin bu adımların hiçbirine takılmadığı anlamına gelinir. Bir sonraki satır okunur dosya sonuna gelinmediyse ilk adıma gidilerek bu süzgeç mantığı tekrarlanır.

**EKRAN ÇIKTILARI ve .LEX İÇERİKLERİ**

**A.FOR**



**A.LEX İÇERİĞİ**

KEYWORD (PROGRAM)

IDENTIFIER (MAIN)

KEYWORD (INTEGER)

IDENTIFIER (N)

IDENTIFIER (X)

KEYWORD (EXTERNAL)

IDENTIFIER (SUB1)

KEYWORD (COMMON)

IDENTIFIER (GLOBALS)

IDENTIFIER (N)

IDENTIFIER (X)

INTEGER CONSTANT (0)

KEYWORD (PRINT)

CHARACTER CONSTANT ('Enter number of repeats')

KEYWORD (READ)

IDENTIFIER (N)

KEYWORD (CALL)

IDENTIFIER (SUB1)

IDENTIFIER (X)

IDENTIFIER (SUB1)

KEYWORD (END)

KEYWORD (SUBROUTINE)

IDENTIFIER (SUB1)

IDENTIFIER (X)

IDENTIFIER (DUMSUB)

KEYWORD (INTEGER)

IDENTIFIER (N)

IDENTIFIER (X)

IDENTIFIER (Y)

KEYWORD (EXTERNAL)

IDENTIFIER (DUMSUB)

KEYWORD (COMMON)

IDENTIFIER (GLOBALS)

IDENTIFIER (N)

KEYWORD (DATA)

IDENTIFIER (Y)

INTEGER CONSTANT (0)

KEYWORD (IF)

IDENTIFIER (X)

OPERATOR LESS THEN (.LT.)IDENTIFIER (N)

KEYWORD (THEN)

IDENTIFIER (X)

IDENTIFIER (X)

ADDITION OR UNARY + (+)

INTEGER CONSTANT (1)

IDENTIFIER (Y)

IDENTIFIER (Y)

ADDITION OR UNARY + (+)

INTEGER CONSTANT (1)

KEYWORD (PRINT)

CHARACTER CONSTANT ('x = ')

IDENTIFIER (X)

CHARACTER CONSTANT (', y = ')

IDENTIFIER (Y)

KEYWORD (CALL)

IDENTIFIER (DUMSUB)

IDENTIFIER (X)

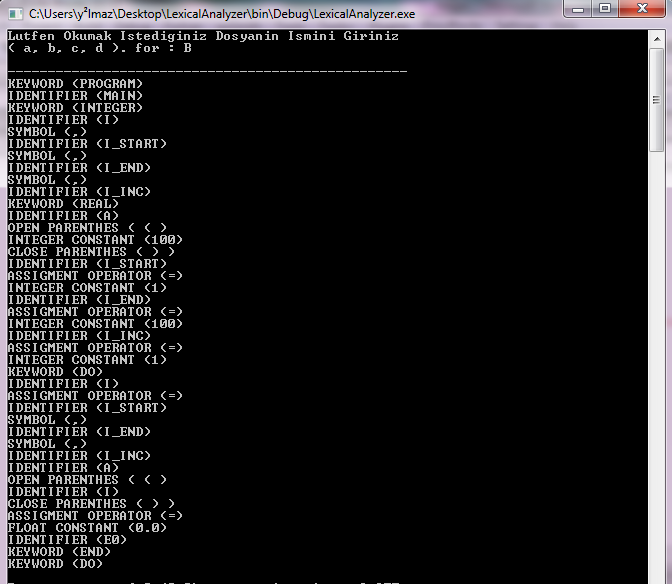
IDENTIFIER (DUMSUB)

KEYWORD (END)

KEYWORD (IF)

KEYWORD (END)

**B.FOR**

****

**KEYWORD (PROGRAM)**

IDENTIFIER (MAIN)

KEYWORD (INTEGER)

IDENTIFIER (I)

IDENTIFIER (I\_START)

IDENTIFIER (I\_END)

IDENTIFIER (I\_INC)

KEYWORD (REAL)

IDENTIFIER (A)

INTEGER CONSTANT (100)

IDENTIFIER (I\_START)

INTEGER CONSTANT (1)

IDENTIFIER (I\_END)

INTEGER CONSTANT (100)

IDENTIFIER (I\_INC)

INTEGER CONSTANT (1)

KEYWORD (DO)

IDENTIFIER (I)

IDENTIFIER (I\_START)

IDENTIFIER (I\_END)

IDENTIFIER (I\_INC)

IDENTIFIER (A)

IDENTIFIER (I)

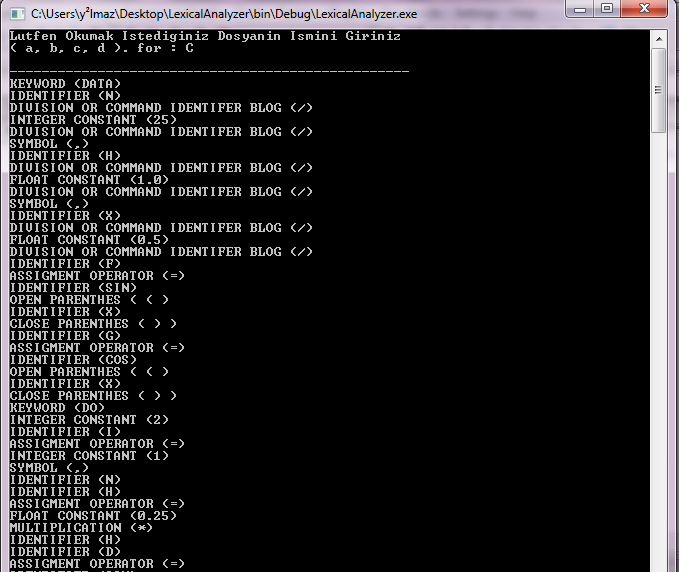
FLOAT CONSTANT (0.0)

IDENTIFIER (E0)

KEYWORD (END)

KEYWORD (DO)

**C.FOR**



KEYWORD (DATA)

IDENTIFIER (N)

INTEGER CONSTANT (25)

IDENTIFIER (H)

FLOAT CONSTANT (1.0)

IDENTIFIER (X)

FLOAT CONSTANT (0.5)

IDENTIFIER (F)

IDENTIFIER (SIN)

IDENTIFIER (X)

IDENTIFIER (G)

IDENTIFIER (COS)

IDENTIFIER (X)

KEYWORD (DO)

INTEGER CONSTANT (2)

IDENTIFIER (I)

INTEGER CONSTANT (1)

IDENTIFIER (N)

IDENTIFIER (H)

FLOAT CONSTANT (0.25)

IDENTIFIER (H)

IDENTIFIER (D)

IDENTIFIER (SIN)

IDENTIFIER (X)

ADDITION OR UNARY + (+)

IDENTIFIER (H)

SUBTRACTION OR UNARY - (-)

IDENTIFIER (F)

IDENTIFIER (Q)

IDENTIFIER (D)

IDENTIFIER (H)

IDENTIFIER (E)

IDENTIFIER (ABS)

IDENTIFIER (G)

SUBTRACTION OR UNARY - (-)

IDENTIFIER (Q)

KEYWORD (PRINT)

IDENTIFIER (H)

IDENTIFIER (D)

IDENTIFIER (Q)

IDENTIFIER (E)

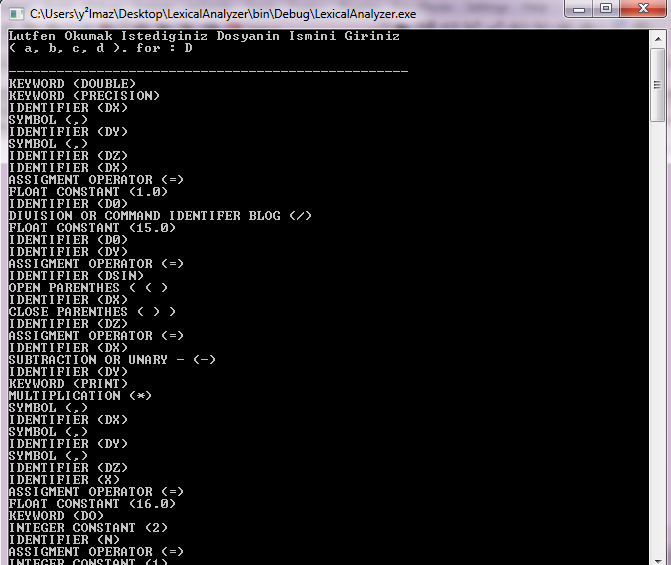
INTEGER CONSTANT (2)

KEYWORD (CONTINUE)

KEYWORD (STOP)

KEYWORD (END)

**D.FOR**

****

KEYWORD (DOUBLE)

KEYWORD (PRECISION)

IDENTIFIER (DX)

IDENTIFIER (DY)

IDENTIFIER (DZ)

IDENTIFIER (DX)

FLOAT CONSTANT (1.0)

IDENTIFIER (D0)

FLOAT CONSTANT (15.0)

IDENTIFIER (D0)

IDENTIFIER (DY)

IDENTIFIER (DSIN)

IDENTIFIER (DX)

IDENTIFIER (DZ)

IDENTIFIER (DX)

SUBTRACTION OR UNARY - (-)

IDENTIFIER (DY)

KEYWORD (PRINT)

IDENTIFIER (DX)

IDENTIFIER (DY)

IDENTIFIER (DZ)

IDENTIFIER (X)

FLOAT CONSTANT (16.0)

KEYWORD (DO)

INTEGER CONSTANT (2)

IDENTIFIER (N)

INTEGER CONSTANT (1)

INTEGER CONSTANT (52)

IDENTIFIER (X)

FLOAT CONSTANT (0.25)

IDENTIFIER (X)

IDENTIFIER (Y)

IDENTIFIER (F)

IDENTIFIER (X)

KEYWORD (PRINT)

IDENTIFIER (N)

IDENTIFIER (X)

IDENTIFIER (Y)

INTEGER CONSTANT (2)

KEYWORD (CONTINUE)

IDENTIFIER (X)

IDENTIFIER (SIN)

FLOAT CONSTANT (1.0)

IDENTIFIER (Y)

IDENTIFIER (F)

FLOAT CONSTANT (1.0)

KEYWORD (PRINT)

IDENTIFIER (X)

IDENTIFIER (Y)

KEYWORD (STOP)

KEYWORD (END)

KEYWORD (FUNCTION)

IDENTIFIER (F)

IDENTIFIER (X)

KEYWORD (IF)

IDENTIFIER (ABS)

IDENTIFIER (X)

OPERATOR GREATER THEN or EQUAL TO (.GE.)FLOAT CONSTANT (1.9)

KEYWORD (THEN)

IDENTIFIER (F)

IDENTIFIER (X)

SUBTRACTION OR UNARY - (-)

IDENTIFIER (SIN)

IDENTIFIER (X)

KEYWORD (ELSE)

IDENTIFIER (T)

IDENTIFIER (X)

INTEGER CONSTANT (3)

FLOAT CONSTANT (6.0)

IDENTIFIER (F)

IDENTIFIER (T)

KEYWORD (DO)

INTEGER CONSTANT (2)

IDENTIFIER (N)

INTEGER CONSTANT (1)

INTEGER CONSTANT (9)

IDENTIFIER (T)

SUBTRACTION OR UNARY - (-)

IDENTIFIER (T)

IDENTIFIER (X)

IDENTIFIER (X)

KEYWORD (REAL)

INTEGER CONSTANT (2)

IDENTIFIER (N)

ADDITION OR UNARY + (+)

INTEGER CONSTANT (2)

INTEGER CONSTANT (2)

IDENTIFIER (N)

ADDITION OR UNARY + (+)

INTEGER CONSTANT (3)

IDENTIFIER (F)

IDENTIFIER (F)

ADDITION OR UNARY + (+)

IDENTIFIER (T)

INTEGER CONSTANT (2)

KEYWORD (CONTINUE)

KEYWORD (END)

KEYWORD (IF)

KEYWORD (RETURN)

KEYWORD (END)

**MAIN.C - SOURCE CODE**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define MAX\_KEYWORD\_LENGTH 18

#define MAX\_COLUMNS\_LENGTH 73

#define MAX\_KEYWORD 100

void fill\_keywords(char keywords[MAX\_KEYWORD][MAX\_KEYWORD\_LENGTH],int \*keyword\_count);

void read\_the\_codes(char name,char keywords[MAX\_KEYWORD][MAX\_KEYWORD\_LENGTH], int keyword\_count);

int IsCommand(char \*row);

int main()

{

char keywords[MAX\_KEYWORD][MAX\_KEYWORD\_LENGTH];

int keyword\_count = 0;

fill\_keywords(keywords,&keyword\_count);

char c;

printf("Lutfen Okumak Istediginiz Dosyanin Ismini Giriniz\n");

printf("( a, b, c, d ). for : ");

scanf("%c",&c);

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

read\_the\_codes(c,keywords,keyword\_count);

return 0;

}

void sifirla(char eski[MAX\_COLUMNS\_LENGTH])

{

int i;

for(i=0;i<MAX\_COLUMNS\_LENGTH;i++)

eski[i]='\0';

}

void read\_the\_codes(char name,char keywords[MAX\_KEYWORD][MAX\_KEYWORD\_LENGTH], int keyword\_count)

{

FILE \*fPtr,\*lPtr;

char file\_name[] = " .for";

file\_name[0]=name;

if( (fPtr=fopen(file\_name,"r"))==NULL ){

printf("Dosyasi Acilamadi");return;

}

file\_name[2]='l';file\_name[3]='e';file\_name[4]='x';

if( (lPtr=fopen(file\_name,"w"))==NULL ){

printf("Dosyasi Acilamadi");return;

}

char currunt\_row[MAX\_COLUMNS\_LENGTH];

fgets(currunt\_row,MAX\_COLUMNS\_LENGTH,fPtr);

int length,i,j,n=0,c=0;

int c\_digit=0, c\_char=0,c\_keyword=0,c\_id=0;

while(!feof(fPtr)){

currunt\_row[MAX\_COLUMNS\_LENGTH]=NULL;

length=strlen(currunt\_row);

currunt\_row[length-1] = '\0';

//Currunt Row is Not Command

if(!IsCommand(currunt\_row)){

c=0;//is carry?

char carry[MAX\_COLUMNS\_LENGTH];

for(i=0;currunt\_row[i]!='\0';i++){

if(c==0 && currunt\_row[i]==' ') continue; //skip while-spaces

fflush(NULL); printf("\b\b\b\b");

if(currunt\_row[i]=='\'' || (c\_char && c)){ //Is Charters Constant

carry[n++]=currunt\_row[i];

if(c && currunt\_row[i]=='\''){

carry[n+1]='\0';

printf("CHARACTER CONSTANT (%s)\n",carry);

fprintf(lPtr,"CHARACTER CONSTANT (%s)\n",carry);

c\_char=0;c=0;n=0;

sifirla(carry);

}

else {c=1; c\_char=1;}

continue;

}

switch(currunt\_row[i]){ //Operators

case '+': printf("ADDITION OR UNARY + (+)\n");

fprintf(lPtr,"ADDITION OR UNARY + (+)\n"); continue;

case '-': printf("SUBTRACTION OR UNARY - (-)\n");

fprintf(lPtr,"SUBTRACTION OR UNARY - (-)\n"); continue;

case '\*': if(currunt\_row[i+1]=='\*') {

printf("EXPONENTIATION (\*\*)\n");

printf(lPtr,"EXPONENTIATION (\*\*)\n");i++; }

else {

printf("MULTIPLICATION (\*)\n");

printf(lPtr,"MULTIPLICATION (\*)\n");}

continue;

case '/': printf("DIVISION OR COMMAND IDENTIFER BLOG (/)\n");

printf(lPtr,"DIVISION OR COMMAND IDENTIFER BLOG (/)\n");continue;

case '(': printf("OPEN PARENTHES ( ( )\n");

printf(lPtr,"OPEN PARENTHES ( ( )\n");continue;

case ')': printf("CLOSE PARENTHES ( ) )\n");

printf(lPtr,"CLOSE PARENTHES ( ) )\n");continue;

case ',': printf("SYMBOL (,)\n");

printf(lPtr,"SYMBOL (,)\n"); continue;

case '=': printf("ASSIGMENT OPERATOR (=)\n");

printf(lPtr,"ASSIGMENT OPERATOR (=)\n");continue;

case ' ': continue;

}

if(isalpha(currunt\_row[i]) || ( c\_id && ( isdigit(currunt\_row[i]) || currunt\_row[i]=='\_') ) ){

carry[n++]=currunt\_row[i]; c\_id=1;c=1;

if(currunt\_row[i+1] == ' ' || currunt\_row[i+1] ==',' || currunt\_row[i+1] =='+' ||

currunt\_row[i+1] =='-' || currunt\_row[i+1] =='\*' || currunt\_row[i+1] =='/' ||

currunt\_row[i+1]=='\0' || currunt\_row[i+1] =='(' || currunt\_row[i+1] ==')' ||

currunt\_row[i+1] ==',' || currunt\_row[i+1] =='=')

{

i+=currunt\_row[i+1]=='\0';

carry[n] = '\0';

}

else continue;

int b; int m=0;

for(b=0;b<keyword\_count;b++){

m=0;

for(j=0;j<n;j++){

if(carry[j]==keywords[b][j]) m++;

}

if(m==n) break;

}

if(m==n && n>1) {

printf("KEYWORD (%s)\n",carry);

fprintf(lPtr,"KEYWORD (%s)\n",carry);

}

else {

printf("IDENTIFIER (%s)\n",carry);

fprintf(lPtr,"IDENTIFIER (%s)\n",carry);

}

c=0;n=0;c\_digit=0;c\_id=0;

sifirla(carry);

continue;

}

if((isdigit(currunt\_row[i]) && (c==0 || c\_digit))

|| (c && c\_digit && currunt\_row[i]=='.' && isdigit(currunt\_row[i+1]) ) ) {

carry[n]=currunt\_row[i];

c=1;n++;c\_digit=1;

if(!((isdigit(currunt\_row[i+1])) || (currunt\_row[i+1]=='.' && isdigit(currunt\_row[i+2]))) ||

currunt\_row[i+1]=='\*' || currunt\_row[i+1]=='/' || currunt\_row[i+1]=='+' || currunt\_row[i+1]=='-'){

int flat=0;

for(j=1;j<n;j++){

if(carry[j]=='.') {flat=1;break;}

}

if(flat) {

printf("FLOAT CONSTANT (%s)\n",carry);

fprintf(lPtr,"FLOAT CONSTANT (%s)\n",carry);

}

else{

printf("INTEGER CONSTANT (%s)\n",carry);

fprintf(lPtr,"INTEGER CONSTANT (%s)\n",carry);

}

c=0;n=0;c\_digit=0;

sifirla(carry);

}

continue;

}

if(currunt\_row[i]=='.'){

if(currunt\_row[i+1]=='E' && currunt\_row[i+2]=='Q' && currunt\_row[i+3]=='.'){

printf("OPERATOR EQUAL TO (.EQ.)\n");

fprintf(lPtr,"OPERATOR EQUAL TO (.EQ.)"); i+=3;

}

else if(currunt\_row[i+1]=='N' && currunt\_row[i+2]=='E' && currunt\_row[i+3]=='.'){

printf("OPERATOR NOT EQUAL TO (.NE.)\n");

fprintf(lPtr,"OPERATOR NOT EQUAL TO (.NE.)"); i+=3;

}

else if(currunt\_row[i+1]=='L' && currunt\_row[i+2]=='T' && currunt\_row[i+3]=='.'){

printf("OPERATOR LESS THEN (.LT.)\n");

fprintf(lPtr,"OPERATOR LESS THEN (.LT.)"); i+=3;

}

else if(currunt\_row[i+1]=='L' && currunt\_row[i+2]=='E' && currunt\_row[i+3]=='.'){

printf("OPERATOR LESS THEN or EQUAL TO (.LE.)\n");

fprintf(lPtr,"OPERATOR LESS THEN or EQUAL TO (.Le.)"); i+=3;

}

else if(currunt\_row[i+1]=='G' && currunt\_row[i+2]=='T' && currunt\_row[i+3]=='.'){

printf("OPERATOR GREATER THEN (.GT.)\n");

fprintf(lPtr,"OPERATOR GREATER THEN (.GT.)"); i+=3;

}

else if(currunt\_row[i+1]=='G' && currunt\_row[i+2]=='E' && currunt\_row[i+3]=='.'){

printf("OPERATOR GREATER THEN or EQUAL TO (.GE.)\n");

fprintf(lPtr,"OPERATOR GREATER THEN or EQUAL TO (.GE.)"); i+=3;

}

else if(currunt\_row[i+1]=='E' && currunt\_row[i+2]=='Q' && currunt\_row[i+3]=='V' && currunt\_row[i+4]=='.'){

printf("OPERATOR EQUVALENT TO (.EQV.)\n");

fprintf(lPtr,"OPERATOR EQUVALENT TO (.EQV.)"); i+=4;

}

else if(currunt\_row[i+1]=='N' && currunt\_row[i+2]=='E' && currunt\_row[i+3]=='Q' &&

currunt\_row[i+4]=='V' && currunt\_row[i+5]=='.'){

printf("OPERATOR NOT EQUVALENT TO (.NEQV.)\n");

fprintf(lPtr,"OPERATOR NOT EQUVALENT TO (.NEQV.)"); i+=5;

}

}

}

//printf("%s\n",currunt\_row);

}

fgets(currunt\_row,MAX\_COLUMNS\_LENGTH,fPtr);

}

fclose(fPtr);

fclose(lPtr);

}

int IsCommand(char \*row)

{

if(strlen(row) ==1){

return row[0]=='C' || row[0]=='\*' ;

}

return (row[0]=='C' || row[0]=='\*') && row[1]==' ';

}

/\* Keyword List \*/

void fill\_keywords(char keywords[MAX\_KEYWORD][MAX\_KEYWORD\_LENGTH],int \*keyword\_count)

{

FILE \*fPtr;

if( (fPtr=fopen("keywords.txt","r"))==NULL ){

printf("keywords.txt dosyasi okunamadi");return;

}

int length;

while(!feof(fPtr)){

fgets(keywords[\*keyword\_count],MAX\_KEYWORD\_LENGTH,fPtr);

length = strlen(keywords[\*keyword\_count]);

keywords[\*keyword\_count][length-1] = '\0';

//printf("%s\n",keywords[\*keyword\_count]);

(\*keyword\_count)++;

}

fclose(fPtr);

}