**LÉXICO**

Con los siguientes cambios en el main.c, el compilador:

1.- Hace el análisis léxico, el análisis sintáctico, el análisis semántico y genera el código

2.- Hace eco (duplica) el programa fuente TlNY hacia el listado junto con los números de línea

3- Presenta la información en cada token a medida que el analizador léxico lo reconoce

MODIFICAR EL main.c

Se debe realizar 3 cambios, los cuales se detallan

#include "globals.h"

#define NO\_PARSE **TRUE** **(1.- Hace el análisis léxico)**

#define NO\_ANALYZE **FALSE**

#define NO\_CODE **FALSE**

#include "util.h"

#if NO\_PARSE

#include "scan.h"

#else

#include "parse.h"

#if !NO\_ANALYZE

#include "analyze.h"

#if !NO\_CODE

#include "cgen.h"

#endif

#endif

#endif

/\* allocate global variables \*/

int lineno = 0;

FILE \* source;

FILE \* listing;

FILE \* code;

/\* allocate and set tracing flags \*/

int EchoSource **= TRUE**; **(2.- Hace eco (duplica) el programa fuente )**

int TraceScan = **TRUE**; **(3.- Presenta la información en cada token)**

int TraceParse = **FALSE**;

int TraceAnalyze = **FALSE**;

int TraceCode = **FALSE**;

int Error = FALSE;

main( int argc, char \* argv[] )

{ TreeNode \* syntaxTree;

char pgm[120]; /\* source code file name \*/

if (argc != 2)

{ fprintf(stderr,"usage: %s <filename>\n",argv[0]);

exit(1);

}

strcpy(pgm,argv[1]) ;

if (strchr (pgm, '.') == NULL)

strcat(pgm,".tny");

source = fopen(pgm,"r");

if (source==NULL)

{ fprintf(stderr,"File %s not found\n",pgm);

exit(1);

}

listing = stdout; /\* send listing to screen \*/

fprintf(listing,"\nTINY COMPILATION: %s\n",pgm);

#if NO\_PARSE

while (getToken()!=ENDFILE);

#else

syntaxTree = parse();

if (TraceParse) {

fprintf(listing,"\nSyntax tree:\n");

printTree(syntaxTree);

}

#if !NO\_ANALYZE

if (! Error)

{ if (TraceAnalyze) fprintf(listing,"\nBuilding Symbol Table...\n");

buildSymtab(syntaxTree);

if (TraceAnalyze) fprintf(listing,"\nChecking Types...\n");

typeCheck(syntaxTree);

if (TraceAnalyze) fprintf(listing,"\nType Checking Finished\n");

}

#if !NO\_CODE

if (! Error)

{ char \* codefile;

int fnlen = strcspn(pgm,".");

codefile = (char \*) calloc(fnlen+4, sizeof(char));

strncpy(codefile,pgm,fnlen);

strcat(codefile,".tm");

code = fopen(codefile,"w");

if (code == NULL)

{ printf("Unable to open %s\n",codefile);

exit(1);

}

codeGen(syntaxTree,codefile);

fclose(code);

}

#endif

#endif

#endif

fclose(source);

return 0;

}

**A CONTINUACIÓN SE PRESENTA EL RESULTADO del archivo LEXICO.TXT después de ejecutar el archivo LEXICO.bat**

TINY COMPILATION: SAMPLE.TNY

1: read x;

1: reserved word: read

1: ID, name= x

1: ;

2: if 0 < x then

2: reserved word: if

2: NUM, val= 0

2: <

2: ID, name= x

2: reserved word: then

3: fact := 1;

3: ID, name= fact

3: :=

3: NUM, val= 1

3: ;

4: repeat

4: reserved word: repeat

5: fact := fact \* x;

5: ID, name= fact

5: :=

5: ID, name= fact

5: \*

5: ID, name= x

5: ;

6: x := x - 1

6: ID, name= x

6: :=

6: ID, name= x

6: -

6: NUM, val= 1

7: until x = 0;

7: reserved word: until

7: ID, name= x

7: =

7: NUM, val= 0

7: ;

8: write fact

8: reserved word: write

8: ID, name= fact

9: end

9: reserved word: end

10: EOF