

ID	Left-hand side		Right-hand side
	<i>program</i>	→	begin <i>statement-list</i> end
	<i>statement-list</i>	→	<i>statement</i>
	<i>statement-list</i>	→	<i>statement-list</i> ; <i>statement</i>
	<i>statement</i>	→	id := <i>expression</i>
	<i>statement</i>	→	read (<i>identifier-list</i>)
	<i>statement</i>	→	write (<i>expression-list</i>)
	<i>identifier-list</i>	→	id
	<i>identifier-list</i>	→	<i>identifier-list</i> , id
	<i>expression-list</i>	→	<i>expression</i>
	<i>expression-list</i>	→	<i>expression-list</i> , <i>expression</i>
	<i>expression</i>	→	<i>primary</i>
	<i>expression</i>	→	<i>expression</i> <i>add-op</i> <i>primary</i>
	<i>primary</i>	→	(<i>expression</i>)
	<i>primary</i>	→	id
	<i>primary</i>	→	intlit
	<i>add-op</i>	→	+
	<i>add-op</i>	→	-

Token	Specification	Token	Specification
ID	(<i>letter</i> <i>_</i>)(<i>letter</i> <i>digit</i> <i>_</i>)*	ASSIGN	:=
BEGAN	begin	COMMA	,
END	end	SEMICOLON	;
READ	read	LPAREN	(
WRITE	write	RPAREN	)
INTLIT	<i>digit</i> +	PLUS	+
		MINUS	-

File mkmcrcr

```
rm mcrlcx.cpp
make -f makemcrr
```

File rmmcrr

```
rm mcrlcx.cpp
rm *.o
rm mcrr
```

File makemcrr

```
#-----
# File makemcrr creates the micro scanner mcrr
#-----
# Author: Thomas R. Turner
# E-Mail: trturner@uco.edu
# Date: March, 2003
#-----
# Copyright March, 2003 by Thomas R. Turner.
# Do not reproduce without permission from Thomas R. Turner.
#-----
# Object files
#-----
obj      =  mcrr.o \
           mcrlcx.o
#-----
# Bind the micro scanner using the linkage editor
#-----
mcrr:     ${obj}
          g++ -o mcrr ${obj} -lm -ll
#-----
# File mcrr.cpp processes command line arguments
#-----
mcrr.o:   mcrr.cpp mcrlcx.h
          g++ -c -g mcrr.cpp
#-----
# File mcrlcx.cpp is the lex-generated scanner
#-----
mcrlcx.cpp: mcrlcx.l mcrlcx.h
            lex mcrlcx.l
            mv lex.yy.c mcrlcx.cpp
#-----
#-----
```

File mcr.cpp

```
//-----  
//File mcr.cpp processes command line arguments and invokes lex (yylex)  
//to find tokens in the input file.  
//-----  
//Author: Thomas R. Turner  
//E-Mail: trturner@uco.edu  
//Date: April, 2004  
//-----  
//Revised January, 2015  
//-----  
//C++ include files  
//-----  
#include <cstdlib>  
#include <cstring>  
#include <iostream>  
#include <fstream>  
#include <stdio>  
#include <string>  
#include <iomanip>  
using namespace std;  
//-----  
//Application include files  
//-----  
#include "mcrlex.h"  
//-----  
//FileException is thrown when a file whose name is given on the command line  
//cannot be opened.  
//-----  
struct FileException {  
    FileException(const char* fn)  
    {   cout << endl;  
        cout << "File " << fn << " cannot be opened.";  
        cout << endl;  
    }  
};  
//-----  
//-----  
//CommandLineError is thrown when too many arguments are put on the  
//command line  
//-----  
struct CommandLineException {  
    CommandLineException(int m,int a)  
    {   cout << endl;  
        cout << "Too many file names on the command line.";  
        cout << endl;  
        cout << "A maximum of " << m << " file names can appear on the command line.";  
        cout << endl;  
    }  
};
```

```

        cout << a << " file names were entered.";
        cout << endl;
        cout << "p01 (<input file name> (<output file name>))";
    }
};
//-----
//Function Title prints a title
//-----
void Title(ostream& o)
{
    o << endl;
    o << setw(15) << "Token Code";
    o << " ";
    o << setw(15) << "Token Name";
    o << " ";
    o << "Token Spelling";
}
//-----
//Function LexMgr processes the input file, calls yylex, the scanner, and
//produces the output file.
//-----
void LexMgr(FILE* i, ostream& o)
{
    static const char* TokenName[]=
        {"EOF"    ,"BEGIN"  ,"END"    ,"READ"   ,"WRITE"
        ,"INTLIT" ,"ID"     ,"ASSIGN" ,"SEMICOLON","COMMA"
        ,"LPAREN" ,"RPAREN" ,"PLUS"   ,"MINUS"  ,"ERROR"
        };
    Lexer L(i);          //Redirect yylex to read file i instead of
                        //the command line

    Title(o);
    for (int t=yylex();t>0;t=yylex()){
        o << endl;
        o << setw(15) << t;
        o << " ";
        o << setw(15) << TokenName[t];
        o << " ";
        o << L.FetchSpelling();
    }
    o << endl;
}

```

```
//-----  
//Function main processes command line arguments  
//-----  
int main(int argc, char* argv[])  
{    try {  
        char ifn[255], ofn[255];  
        switch (argc) {  
            case 1://no files on the command line  
                cout << "Enter the input file name. ";  
                cin >> ifn;  
                cout << "Enter the output file name. ";  
                cin >> ofn;  
                break;  
            case 2://input file on the command line/prompt for output file  
                strcpy(ifn,argv[1]);  
                cout << "Enter the output file name. ";  
                cin >> ofn;  
                break;  
            case 3://Both files on the input line  
                strcpy(ifn,argv[1]);  
                strcpy(ofn,argv[2]);  
                break;  
            default:  
                throw CommandLineException(2,argc-1);  
                break;  
        }  
        FILE* i=fopen(ifn,"r"); if (!i) throw FileNotFoundException(ifn);  
        ofstream o(ofn); if (!o) throw FileNotFoundException(ofn);  
        LexMgr(i,o);  
        fclose(i);  
        o.close();  
    } catch ( ... ) {  
        cout << endl;  
        cout << "Program Terminated!";  
        cout << endl;  
        cout << "I won't be back!";  
        cout << endl;  
        exit(EXIT_FAILURE);  
    }  
    return 0;  
}
```

File mcrlex.h

```
#ifndef mcrlex_h
#define mcrlex_h 1
//-----
// File mcrlex.h defines class Lexer.
//-----
// Author: Thomas R. Turner
// E-Mail: trturner.uco.edu
// Date: March, 2003
//-----
// Revised January, 2015
//-----
// Copyright March, 2003 by Thomas R. Turner
// Do not reproduce without permission from Thomas R. Turner.
//-----
//-----
// Standard C and C++ include files
//-----
#include <stdio>
#include <fstream>
#include <iostream>
using namespace std;
//-----
//Token definitions
//-----
#include "y.tab.h"
//-----
//Function: yylex
//Function yylex is the mcrner. Function yylex returns an integer
//token code as defined above or 0 if end-of-file has been
//reached.
//-----
#ifdef __cplusplus
extern "C"
#endif
int yylex (void);
//-----
//Class Lexer defines the attributes of a Scanner
//-----
class Lexer {
    int tokencode;           //Code for the most recent token found
public:
    Lexer(FILE* i);          //Constructor used to redirect the keyboard
                             //(stdin) to file i.
    int Lex(void);           //Call the scanner yylex and return the code
                             //found by yylex
    int FetchTokenCode(void); //Return the code of the most recent token
}
```

```
void StoreTokenCode(int T); //Store the token code.  
char* FetchSpelling(void); //Return the spelling of the most recent  
                             //token  
};  
#endif
```

File y.tab.h

```
#ifndef y_tab_h  
#define y_tab_h 1  
//-----  
//Token definitions  
//-----  
#define BEGAN      1  
#define END        2  
#define READ       3  
#define WRITE      4  
#define INTLIT     5  
#define ID         6  
#define ASSIGN     7  
#define SEMICOLON  8  
#define COMMA      9  
#define LPAREN     10  
#define RPAREN     11  
#define PLUS       12  
#define MINUS      13  
#define ERROR      14  
#endif
```

File mcrlex.l

```
%{
//-----
// File mcrlex.l defines a prototype scanner for the micro language.
// The scanner definition is a lex specification.
//-----
// Author: Thomas R. Turner
// E-Mail: trturner@ucok.edu
// Date: March, 2003
//-----
//Copyright March, 2003 by Thomas R. Turner.
//Do not reproduce without permission from Thomas R. Turner
//-----
//-----
// Standard C and C++ Library Include Files
//-----
#include <string>
#include <iostream>
#include <fstream>
#include <cstdio>
//-----
// Application Includes
//-----
#include "mcrlex.h"
//-----
//Token definitions
//-----
#include "y.tab.h"
//-----
//Namespaces
//-----
using namespace std;
//-----
//Externals
//-----
//-----
//Global Variables
//-----
int TokenMgr(int t);           //Token post processing
%}
```



```

%%
[ \t\n]+          ;
[_a-zA-Z][_a-zA-Z0-9]*      return TokenMgr(ID);
[0-9]+            return TokenMgr(INTLIT);
":="             return TokenMgr(ASSIGN);
";"             return TokenMgr(SEMICOLON);
","             return TokenMgr(COMMA);
"("             return TokenMgr(LPAREN);
")"             return TokenMgr(RPAREN);
"+"             return TokenMgr(PLUS);
"-"             return TokenMgr(MINUS);
"."             return TokenMgr(ERROR);
%%
//-----
//Class Lexer implementation
//-----
//Function TokenMgr processes the token after it has been recognized
//-----
int TokenMgr(int t)
{
    if (t!=ID) return t;
    if ((string)yytext=="begin") return BEGAN;
    if ((string)yytext=="end") return END;
    if ((string)yytext=="read") return READ;
    if ((string)yytext=="write") return WRITE;
    return ID;
}
//-----
//Constructor Lexer is used to redirect the input file stream from the
//keyboard to input file stream i.
//-----
Lexer::Lexer(FILE* i)
{
    yyin=i;
}
//-----
//Function Lex calls yylex
//-----
int Lexer::Lex(void)
{
    tokencode=yylex();
    return tokencode;
}

```

```
//-----
//Function FetchSpelling returns a pointer to the spelling of the most
//recent token.
//-----
char* Lexer::FetchSpelling(void)
{
    return (char*)yytext;
}
//-----
//Function FetchTokenCode returns the code of the most recent token
//-----
int Lexer::FetchTokenCode(void)
{
    return tokencode;
}
//-----
//Function StoreTokenCode records the most recent token code
//-----
void Lexer::StoreTokenCode(int T)
{
    tokencode=T;
}
//-----End of Lex Definition-----
```

File t00.mcr

begin read(x); x:=x+2; y:=x-3; write(x,y); end

File t00.trc

Token Code	Token Name	Token Spelling
1	BEGIN	begin
3	READ	read
10	LPAREN	(
6	ID	x
11	RPAREN	)
8	SEMICOLON	;
6	ID	x
7	ASSIGN	:=
6	ID	x
12	PLUS	+
5	INTLIT	2
8	SEMICOLON	;
6	ID	y
7	ASSIGN	:=
6	ID	x
13	MINUS	-
5	INTLIT	3
8	SEMICOLON	;
4	WRITE	write
10	LPAREN	(

6	ID	x
9	COMMA	,
6	ID	y
11	RPAREN	)
8	SEMICOLON	;
2	END	end

~