```
%{
//-------------------------------------------------------------------
// 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----------------------
```