lex Examples(1)

```
letter      [A-Za-z]
digit       [0-9]

%%

begin                 {printf("reserved word begin found\n");}
end                   {printf("reserved word end found\n");}
{letter}{letter}*     {printf("identifier found:%lu:%s\n", yyleng, yytext);}
{digit}{digit}*       {printf("integer found:%lu:%s\n”, yyleng, yytext);}


%%
```

More lex Examples(2)

```
letter      [A-Za-z]
digit       [0-9]

%%

begin                   {return 0; }
end                     {return 1;}
{letter}{letter}*       {return 2;}
{digit}{digit}*         {return 3;}
\n                      {return 4;}

%%
```

More lex Examples(2)

```
Int main() {
    int x;
    while(1) {
      x = yylex();
      printf("%d\n", x);
    }
}
Int yywrap() {
    printf("all done\n");
    exit(0);
}
```

More lex Examples(3)
microTokens.h

```
#define begin 14
#define end 1
#define read 2
#define write 3
#define id 4
#define intliteral 5
#define lparen 6
#define rparen 7
#define semicolon 8
#define comma 9
#define assignop 10
#define plusop 11
#define minusop 12
```

More lex Examples(3)

```
%{
#include "microTokens.h"
%}

letter      [A-Za-z]
digit       [0-9]
```

More lex Examples(3)

```
%%

begin                {return begin;}
end                  {return end;}
read                 {return read;}
write                {return write;}
{letter}{letter}*          {return id;}
{digit}{digit}*      {return intliteral;}
\(                   {return lparen;}
\)                   {return rparen; }
;                    {return semicolon;}
,                    {return comma;}
:=                   {return assignop;}
\+                   {return plusop;}
-                    {return minusop;}
[ \t\n]              {}

%%
```

More lex Examples(3)

```
main() {
    int x;
    while ((x = yylex()))
      printf("Token type of %s is %d\n", yytext, x);
    exit(0);
}
```

More lex Examples(4)
scan.h

```
#define  VAR      1
#define  FLOAT    2
#define  INT      3
#define  MULT     4
#define  DIV      5
#define  PLUS     6
#define  SUB      7
#define  ASSIGN   8
#define  EOL      9
#define  ERROR    10
```

More lex Examples(4)

```
%{
    #include "scan.h"
    #include "IOMngr.h"
    #define YY_INPUT(buf,result,max_size) \
    { int c = getNextSourceChar(); \
        result = (c == EOF) ? YY_NULL : (buf[0] = c, 1); \
      }

%}
letter [A-Za-z]
digit [0-9]
```

More lex Examples(4)

```
%%
{letter}+            {return VAR;}
{digit}+\.{digit}+   {return FLOAT;}
{digit}+             {return INT;}
\*                   {return MULT;}
\/                   {return DIV;}
\+                   {return PLUS;}
\-                   {return SUB;}
\=                   {return ASSIGN;}
\n                   {return EOL;}
\t                   {}
[ ]                  {}
.                    { writeIndicator(getCurrentColumnNum());
                         writeMessage("Illegal Character");
                         return ERROR; }
%%

int yywrap() {
        return 1;
}
```

lex File Format

%{
    C Code
%}

Regular expression (RE) definitions

name   RE

%%

Actions associated with each regular expression

RE Action

%%

C Code

Build lex Program

- flex lex1.l
- This produces a file called lex.yy.c
- gcc -o lex1 lex.yy.c -ll