Sonoma State University
Computer Science Department
CS 460 – Fall 2023 – Watts

Semester Project – Part 1 (aka Project 1)

Due date: Monday, 16 October 2023; 6:59 am.

For this part of the project you will be writing a lexical analyzer. Your program should be written in C++ using the Object Oriented Paradigm.

Specifications

1. Your lexical analyzer should recognize lexemes for identifiers, numeric literals and the symbols listed in the table on the second page of this document.
2. Symbols must start with a letter and may be followed by any number of letters, digits, or underscores.
3. Before writing any code, you should design a regular expression that describes the set of valid lexemes and a DFA that recognizes valid lexemes.
4. Your lexical analyzer should be table driven. The table can be hard coded or stored in an external file.
5. Your analyzer will read a source file (.pl460) and produce 3 files:
   a listing (.lst) file
   a token (.p1) file
   a debugging(.dbg) file
   The listing file should contain the lines of the input file with line numbers and errors. The token file should contain a list of the tokens identified in the file ad their corresponding lexemes.
   The debugging file is for your use. I suggest that it also contain the lines of the input file with line numbers and errors AND, following each line, a list of the tokens found on the line and their corresponding lexemes. Sample input and output files are shown on the last page of this document.
6. Your analyzer should provide a header file containing an enumerated type called token and prototypes for functions available to other parts of your project:

```
/* The GetToken function returns the member of the enumerated
type associated with the next lexeme in the source file. This
function will read lines of source code from the source (.pl460) file as
needed and will echo the lines in the listing (.lst) file. This function
will create a list of the tokens and lexemes in the tokens file (.p1).
This function will make calls to the ReportError function as needed. */
token_type GetToken();

/* The GetLexeme function returns the most recently recognized
lexeme from the source file. GetLexeme can only be called after
GetToken has been called at least once. GetLexeme cannot be
called after the end of the source file has been detected. */
string GetLexeme();

/* The Get_Token_Name function returns a string containing the
name of the token passed to it. */
string GetTokenName(token_type token);

/* The ReportError function will be used by the lexical
analyzer and other parts of your project to report an error.
The lexical analyzer will print the error message following the
line of source code in which it was detected. */
void ReportError (const string & message);
```

7. A framework for this part of the project can be found in the folder Project1Framework in the course pickup folder.

Submission: A tarred and zipped directory containing the files needed to compile and execute your lexical analyzer and a file called README.txt. The directory should be called *lastname*P1 and the tarred and zipped file should be called *lastname*P1.tgz. Your directory should contain a makefile with a default target of P1.out Due date: Monday, 16 October 2023; 6:59 am.

Identifier	α(α|#|_)*	IDENT_T
Numeric Literals	(+|-|λ) (#⁺ | #*.#⁺ | #⁺.#* | #⁺/#⁺)	NUMLIT_T
String Literals	" . . . "	STRLIT_T
Key words	(cad*r) | (cd⁺r) | (cd*ar)	LISTOP_T
	cons	CONS_T
	if	IF_T
	cond	COND_T
	else	ELSE_T
	display	DISPLAY_T
	newline	NEWLINE_T
	and	AND_T
	or	OR_T
	not	NOT_T
	define	DEFINE_T
	let	LET_T
	read	READ_T
Predicates	number?	NUMBERP_T
	list?	LISTP_T
	zero?	ZEROP_T
	null?	NULLP_T
	eof?	EOFP_T
Arithmetic	+	PLUS_T
	-	MINUS_T
	/	DIV_T
	*	MULT_T
	modulo	MODULO_T
	round	ROUND_T
Logical/Relational	=	EQUALTO_T
	>	GT_T
	<	LT_T
	>=	GTE_T
	<=	LTE_T
Other	(	LPAREN_T
	)	RPAREN_T
	'	SQUOTE_T
		ERROR_T
		EOF_T

Sample Input file (P1-1.pl460)

```
identifier ident_1 function
0 123 1.1 -1.1 +1.1 12. -.1
1 12 345 -12 -345 +12 +345
1.12 3.45 -12. -3.45 +1.2 +.345
1/12 3/45 -1/2 -34/5 +1/2 +3/45
"Hello World"

cons if cond display newline and or not define
car cdr cadr cddr ?number
number?  list?$zero?  null?  string?

+ - / * modulo round
= > < >= <=
( ) 'a
```

Sample Listing file (P1-1.lst)

```
Input file: P1-1.pl460
   1: identifier ident_1 function
   2: 0 123 1.1 -1.1 +1.1 12. -.1
   3: 1 12 345 -12 -345 +12 +345
   4: 1.12 3.45 -12. -3.45 +1.2 +.345
   5: 1/12 3/45 -1/2 -34/5 +1/2 +3/45
   6: "Hello World"
   7:
   8: cons if cond display newline and or not define
   9: car cdr cadr cddr ?number
Error at 9,19: Unexpected '?' found
  10: number?  list?$zero?  null?  string?
Error at 10,15: Unexpected '$' found
Error at 10,36: Unexpected '?' found
  11:
  12: + - / * modulo round
  13: = > < >= <=
  14: ( ) 'a
  15:
3 errors found in input file
```

Sample Token file (P1-1.p1)

```
IDENT_T         identifier
IDENT_T         ident_1
IDENT_T         function
NUMLIT_T        0
NUMLIT_T        123
NUMLIT_T        1.1
NUMLIT_T        -1.1
NUMLIT_T        +1.1
NUMLIT_T        12.
```

```
NUMLIT_T        -.1
NUMLIT_T        1
NUMLIT_T        12
NUMLIT_T        345
NUMLIT_T        -12
NUMLIT_T        -345
NUMLIT_T        +12
NUMLIT_T        +345
NUMLIT_T        1.12
NUMLIT_T        3.45
NUMLIT_T        -12.
NUMLIT_T        -3.45
NUMLIT_T        +1.2
NUMLIT_T        +.345
NUMLIT_T        1/12
NUMLIT_T        3/45
NUMLIT_T        -1/2
NUMLIT_T        -34/5
NUMLIT_T        +1/2
NUMLIT_T        +3/45
STRLIT_T        "Hello World"
LISTOP2_T       cons
IF_T            if
COND_T          cond
DISPLAY_T       display
NEWLINE_T       newline
AND_T           and
OR_T            or
NOT_T           not
DEFINE_T        define
LISTOP1_T       car
LISTOP1_T       cdr
LISTOP1_T       cadr
LISTOP1_T       cddr
ERROR_T         ?
IDENT_T         number
NUMBERP_T       number?
LISTP_T         list?
ERROR_T         $
ZEROP_T         zero?
NULLP_T         null?
IDENT_T         string
ERROR_T         ?
PLUS_T          +
MINUS_T         -
DIV_T           /
MULT_T          *
MODULO_T        modulo
ROUND_T         round
EQUALTO_T       =
GT_T            >
LT_T            <
GTE_T           >=
LTE_T           <=
LPAREN_T        (
RPAREN_T        )
SQUOTE_T        '
IDENT_T         a
EOF_T
```

Sample Input file (P1-2.pl460)

```
identifierident_1function
01231.1-1.1+1.112.-.1
112 345-12-345+12+345
1.123.45-12.-3.45+1.2+.345
1/123/45-1/2-34/5+1/2+3/45
"HelloWorld"

consifconddisplaynewlineandornotdefine
carcdrcadrcddr?number
number?list?$zero?null?string?

+-/*moduloround
=><>=<=
()'a
```

Sample Listing file (P1-2.lst)

```
Input file: P1-2.pl460
   1: identifierident_1function
   2: 01231.1-1.1+1.112.-.1
Error at 2,18: Unexpected '.' found
   3: 112 345-12-345+12+345
   4: 1.123.45-12.-3.45+1.2+.345
   5: 1/123/45-1/2-34/5+1/2+3/45
   6: "HelloWorld"
   7:
   8: consifconddisplaynewlineandornotdefine
   9: carcdrcadrcddr?number
Error at 9,15: Unexpected '?' found
  10: number?list?$zero?null?string?
Error at 10,13: Unexpected '$' found
Error at 10,30: Unexpected '?' found
  11:
  12: +-/*moduloround
  13: =><>=<=
  14: ()'a
  15:
4 errors found in input file
```

Sample Token file (P1-2.p1) on next page

Sample Token file (P1-2.p1)

```
IDENT_T         identifierident_1function
NUMLIT_T        01231.1
NUMLIT_T        -1.1
NUMLIT_T        +1.112
ERROR_T         .
NUMLIT_T        -.1
NUMLIT_T        112
NUMLIT_T        345
NUMLIT_T        -12
NUMLIT_T        -345
NUMLIT_T        +12
NUMLIT_T        +345
NUMLIT_T        1.123
NUMLIT_T        .45
NUMLIT_T        -12.
NUMLIT_T        -3.45
NUMLIT_T        +1.2
NUMLIT_T        +.345
NUMLIT_T        1/123
DIV_T           /
NUMLIT_T        45
NUMLIT_T        -1/2
NUMLIT_T        -34/5
NUMLIT_T        +1/2
NUMLIT_T        +3/45
STRLIT_T        "HelloWorld"
IDENT_T         consifconddisplaynewlineandornotdefine
IDENT_T         carcdrcadrcddr
ERROR_T         ?
IDENT_T         number
NUMBERP_T       number?
LISTP_T         list?
ERROR_T         $
ZEROP_T         zero?
NULLP_T         null?
IDENT_T         string
ERROR_T         ?
PLUS_T          +
MINUS_T         -
DIV_T           /
MULT_T          *
IDENT_T         moduloround
EQUALTO_T       =
GT_T            >
LT_T            <
GTE_T           >=
LTE_T           <=
LPAREN_T        (
RPAREN_T        )
SQUOTE_T        '
IDENT_T         a
 EOF_T
```