

PONTIFICIA UNIVERSIDAD CATOLICA MADRE Y MAESTRA



Nombre:

Félix Alejandro Guzmán 2014 - 0565

Materia:

ST-ISC-314-T-001 Programación 3

Profesor:

Juan R. Núñez P.

Practica sobre:

Práctica 3 - Parser

Fecha de Entrega:

martes, 27 de febrero de 2018

Tiger.lex

```
type sval = Tokens.sval
type pos = int
type ('a,'b) token = ('a,'b) Tokens.token
type lexresult = (sval,pos) token
val lineNum = ErrorMsg.lineNum
val linePos = ErrorMsg.linePos
val strBuilder = ref ""
val strPosition = ref 0
val unclosedStr = ref false
val cmCount = ref 0
fun eof() =
  let
    val pos = hd(!linePos)
  in
    if !cmCount > 0
    then (ErrorMsg.error pos (Int.toString(!cmCount) ^ " unclosed
comments "); cmCount := 0; Tokens.EOF(pos,pos))
    else if !unclosedStr = true
    then (ErrorMsg.error pos ("unclosed string starting " ^
Int.toString(!strPosition)); Tokens.EOF(pos,pos))
    else (Tokens.EOF(pos,pos))
  end

%%
digit  = [0-9];
letter = [a-zA-Z];
%header (functor TigerLexFun(structure Tokens: Tiger_TOKENS));
%s COMMENT NPSTRING STRING;
%%
<INITIAL>\n => (lineNum := !lineNum+1; linePos := yypos+1 :: !linePos;
continue());
<INITIAL>[\ \t] => (continue());

<INITIAL>type  => (Tokens.TYPE(yypos,yypos+4));
<INITIAL>var   => (Tokens.VAR(yypos,yypos+3));
<INITIAL>function => (Tokens.FUNCTION(yypos,yypos+8));
<INITIAL>break => (Tokens.BREAK(yypos,yypos+5));
<INITIAL>of    => (Tokens.OF(yypos,yypos+2));
<INITIAL>end   => (Tokens.END(yypos,yypos+3));
<INITIAL>in    => (Tokens.IN(yypos,yypos+2));
<INITIAL>nil   => (Tokens.NIL(yypos,yypos+3));
<INITIAL>let   => (Tokens.LET(yypos,yypos+3));
<INITIAL>do    => (Tokens.DO(yypos,yypos+2));
<INITIAL>to    => (Tokens.TO(yypos,yypos+2));
<INITIAL>for   => (Tokens.FOR(yypos,yypos+3));
<INITIAL>while => (Tokens.WHILE(yypos,yypos+5));
```

```

<INITIAL>else    => (Tokens.ELSE(yypos,yypos+4));
<INITIAL>then    => (Tokens.THEN(yypos,yypos+4));

<INITIAL>if => (Tokens.IF(yypos, yypos+2));
<INITIAL>array  => (Tokens.ARRAY(yypos, yypos+5));
<INITIAL>":="    => (Tokens.ASSIGN(yypos, yypos+2));
<INITIAL>"|"     => (Tokens.OR(yypos, yypos+1));
<INITIAL>"&"     => (Tokens.AND(yypos, yypos+1));
<INITIAL>">="    => (Tokens.GE(yypos, yypos+2));
<INITIAL>">"     => (Tokens.GT(yypos, yypos+1));
<INITIAL>"<="    => (Tokens.LE(yypos, yypos+2));
<INITIAL>"<"     => (Tokens.LT(yypos, yypos+1));
<INITIAL>"<>"    => (Tokens.NEQ(yypos, yypos+2));
<INITIAL>"="     => (Tokens.EQ(yypos, yypos+1));
<INITIAL>"/"     => (Tokens.DIVIDE(yypos, yypos+1));
<INITIAL>"*"     => (Tokens.TIMES(yypos, yypos+1));
<INITIAL>"-"     => (Tokens.MINUS(yypos, yypos+1));
<INITIAL>"+ "    => (Tokens.PLUS(yypos, yypos+1));
<INITIAL>"."     => (Tokens.DOT(yypos, yypos+1));

<INITIAL>"}"     => (Tokens.RBRACE(yypos, yypos+1));
<INITIAL>"{"     => (Tokens.LBRACE(yypos, yypos+1));
<INITIAL>"]"     => (Tokens.RBRACK(yypos, yypos+1));
<INITIAL>"["     => (Tokens.LBRACK(yypos, yypos+1));
<INITIAL>")"     => (Tokens.RPAREN(yypos, yypos+1));
<INITIAL>"("     => (Tokens.LPAREN(yypos, yypos+1));
<INITIAL>";"     => (Tokens.SEMICOLON(yypos, yypos+1));
<INITIAL>":"     => (Tokens.COLON(yypos, yypos+1));
<INITIAL>","     => (Tokens.COMMA(yypos, yypos+1));

<INITIAL>{digit}+    => (Tokens.INT(valOf(Int.fromString yytext), yypos, yypos
+ size yytext));
<INITIAL>{letter}+({letter}|{digit}|_)* => (Tokens.ID(yytext, yypos, yypos +
size yytext));

<INITIAL>\"        => (YYBEGIN STRING; strBuilder := ""; strPosition := yypos;
uncloseStr := true; continue());
<STRING>\"         => (YYBEGIN INITIAL; uncloseStr := false;
Tokens.STRING(!strBuilder, !strPosition, yypos+1));
<STRING>\\(n|t|\\^c|[0-9]{3}|\\\"|\\\) => (strBuilder := !strBuilder ^
valOf(String.fromString yytext); continue());
<STRING>[\\]       => (YYBEGIN NPSTRING; continue());
<NPSTRING>[\\n]    => (lineNum := !lineNum+1; linePos := yypos+1 :: !linePos;
continue());
<NPSTRING>[\\ \\t\\f] => (continue());
<NPSTRING>[\\]     => (YYBEGIN STRING; continue());
<NPSTRING>\\.      => (ErrorMsg.error yypos ("Illegal escape character: " ^
yytext); YYBEGIN STRING; continue());

```

```

<STRING>[\n]    => (lineNum := !lineNum+1; linePos := yypos+1 :: !linePos;
ErrorMsg.error yypos ("illegal linebreak in string literal "); continue());
<STRING>.\       => (strBuilder := !strBuilder ^ yytext; continue());

<INITIAL>"/*"    => (YYBEGIN COMMENT; cmCount := !cmCount + 1; continue());
<COMMENT>"/*"    => (cmCount := !cmCount + 1; continue());
<COMMENT>"/"     => (cmCount := !cmCount - 1; if !cmCount = 0 then YYBEGIN
INITIAL else ()); continue());
<COMMENT>[\n]    => (lineNum := !lineNum+1; linePos := yypos+1 :: !linePos;
continue());
<COMMENT>.\      => (continue());

.               => (ErrorMsg.error yypos ("Illegal character: " ^ yytext);
continue());

```

Sources.cm

Group is

```

tiger.lex
errormsg.sml
parsetest.sml
tiger.grm

```

```

$/smlnj-lib.cm
$/ml-yacc-lib.cm
$/basis.cm

```

Factorial

```
/* define a recursive function */
```

```
let
```

```
/* calculate n! */
```

```
function nfactor(n: int): int =
```

```
    if n = 0
```

```
        then 1
```

```
    else n * nfactor(n-1)
```

```
in
```

```
    nfactor(10)
```

```
end
```



```
factorial.tig - Tarea 3 - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
sources.cm factorial.tig x
1 /* define a recursive function */
2 let
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: run.x86-win: + - v x
PS C:\Users\felix\Documents\Programacion 3\Tarea 3> sml -m sources.cm
Standard ML of New Jersey v110.82 [built: Wed Oct 25 10:36:05 2017]
[scanning sources.cm]
[attempting to load plugin $/lex-ext.cm]
[library $/lex-ext.cm is stable]
[library $smlnj/cm/tools.cm is stable]
[library $smlnj/internal/cm-lib.cm is stable]
[library $SMLNJ-BASIS/basis.cm is stable]
[library $SMLNJ-BASIS/(basis.cm):basis-common.cm is stable]
[plugin $/lex-ext.cm loaded successfully]
[attempting to load plugin $/mllex-tool.cm]
[library $/mllex-tool.cm is stable]
[plugin $/mllex-tool.cm loaded successfully]
[attempting to load plugin $/grm-ext.cm]
[library $/grm-ext.cm is stable]
[plugin $/grm-ext.cm loaded successfully]
[attempting to load plugin $/mlyacc-tool.cm]
[library $/mlyacc-tool.cm is stable]
[plugin $/mlyacc-tool.cm loaded successfully]
[library $/ml-yacc-lib.cm is stable]
[library $SMLNJ-ML-YACC-LIB/ml-yacc-lib.cm is stable]
binfile format error: bad magic number
[compiling (sources.cm):tiger-grm.sig]
[code: 56, env: 978 bytes]
binfile format error: bad magic number
[compiling (sources.cm):errormsg.sml]
[code: 2680, data: 156, env: 451 bytes]
binfile format error: bad magic number
[compiling (sources.cm):tiger.lex.sml]
[code: 133411, data: 607, env: 3046 bytes]
binfile format error: bad magic number
[compiling (sources.cm):tiger-grm.sml]
[code: 54316, data: 4360, env: 1144 bytes]
binfile format error: bad magic number
[compiling (sources.cm):parsetest.sml]
[code: 5753, data: 93, env: 133 bytes]
[New bindings added.]
- Parse.parse "factorial.tig";
val it = () : unit
- |
```

Test.tig

of let

```
/* BELOW TEST THE STRING RELATED ERRORS*/
```

```
"ld\j\
```

```
\kfd\\asli\n\"k\t\123j^Gsd\
```

```
\\      \f"
```

```
/*BELOW TEST THE GENERAL CASES*/
```

It

```
var if8 = 644
```

```
var xt = 01
```

```
va s_4 = 0
```

```
var sss____44__ss_0=1;
```

```
var T5_ = 7
```

```
type myintT6 = int
```

```
type arrtype = array of myintT6
```

```
var arr1:arrtype := arrtype [10] of 0
```

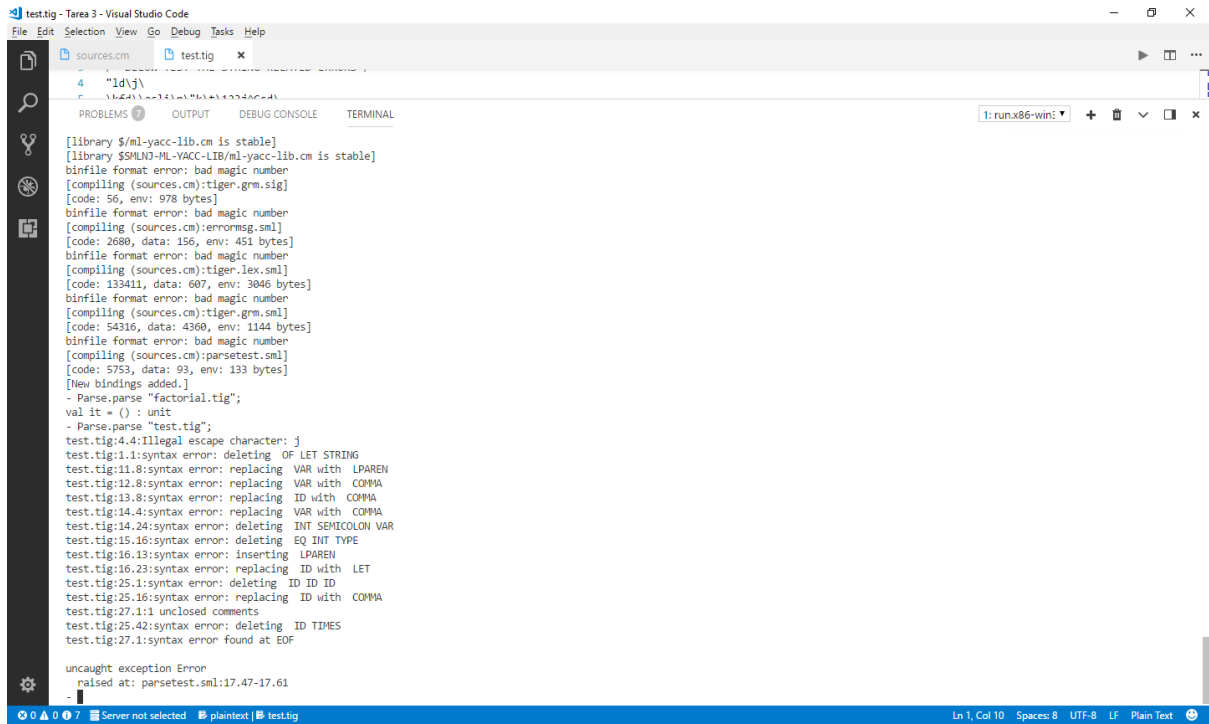
in

```
arr1
```

end

```
BELOW TEST THE NESTED COMMENT & UNCLOSED COMMENT*/
```

```
/* arr1 is valid since expression 0 is int = myint /
```



Queens.tig

/ A program to solve the 8-queens problem */

let

var N = 8

type intArray = array of int

var row := intArray [N] of 0

var col := intArray [N] of 0

var diag1 := intArray [N+N-1] of 0

var diag2 := intArray [N+N-1] of 0

function printtoard() =

(for i := 0 to N-1

do (for j := 0 to N-1

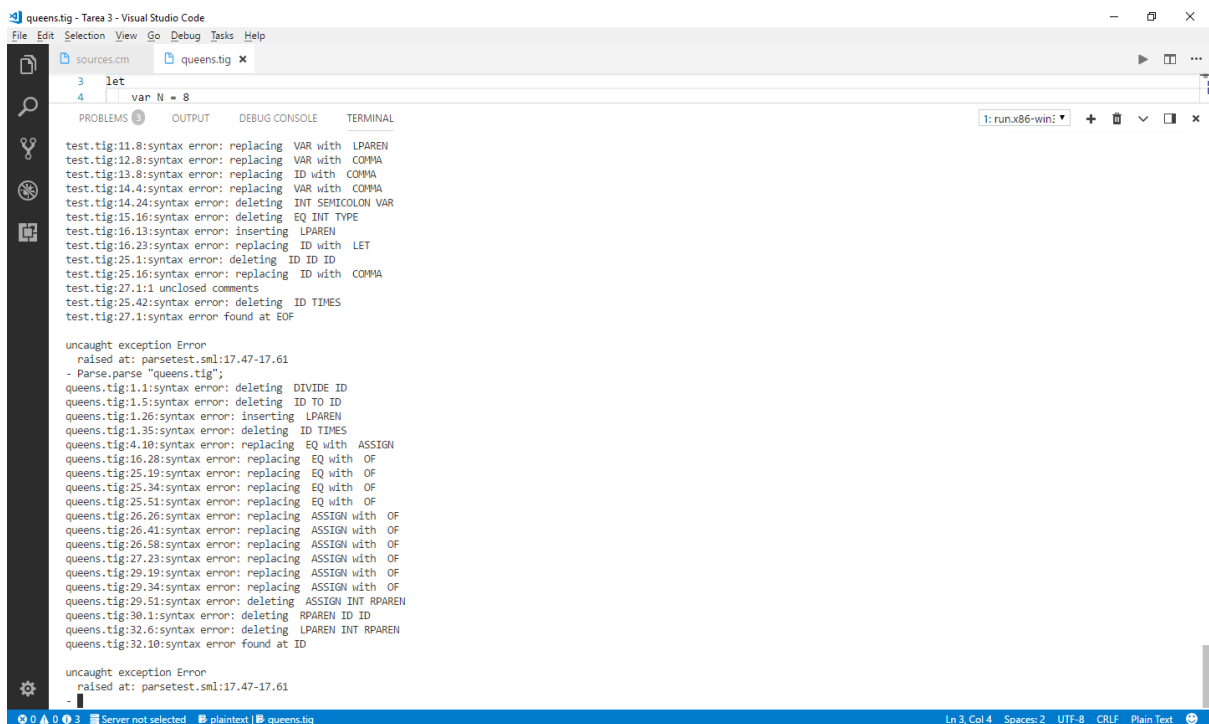
do print(if col[i]=j then " O" else " .");

```

        print("\n"));
        print("\n"))

function try(c:nt) =
( /* for i:= 0 to c do prit("."); print("\n"); flush(); */
    if c=N
    then printboard()
    else for r := 0 to N-1
        do if row[r]=0 & diag1[r+c]=0 & diag2[r+7-c]=0
            then (rw[r]:=1; diag1[r+c]:=1; diag2[r+7-c]:=1;
                col[c]:=r;
                try(c+1);
                row[r]:=0; diag1[r+c]:=0; diag2[r+7-c]:=0)
        )
    i try(0)
en

```



En esta práctica se implemento un parser para el lenguaje Tiger utilizando algunos archivos como: `errmsg.sml`, `parsetest.sml`, `sources.cm`, `tiger.grm`, y `tiger.lex.sml`.

En el primer ejemplo no hay error de salida, mientras que en el segundo y tercero si hay.