Data Structures **Project p07**
CS 3613 **Pascal Scanner**

Project: Program **7** reads a file containing a Pascal program and produces a list of tokens in the file. Use a perfect hash to lookup reserve words in the Pascal grammar.

Program Files:

File	**Description**
pas.cpp	File **pas.cpp** contains functions that process command line arguments and direct lexical analysis.
paslex.h	File **paslex.h** defines the interface to functions defined in file **paslex.l**.
paslex.l	File **paslex.l** specifies the Pascal scanner using regular expressions intermixed with functions and other definitions. File **paslex.l** is translated by the UNIX utility ***lex*** into a C source file. The C++ compiler is invoked to translate the output of ***lex*** to a C++ object file compatible with other C++ objects. C-functions defined in file **paslex.l** employ a perfect hash.
pashash.h	File **pashash.h** defines class *Hash*. **class** *Hash* defines member data and functions for a hash.
pashash.cpp	File **pashash.cpp** implements **class** *Hash*.
makepas	File **makepas** contains instructions for program **pas**. Instructions are written for the UNIX utility ***make***. Program **pas** is contained in file **pas**.

Command Line: Project **7** can be invoked with zero or one or two program parameters. The first program parameter is the input file name. The second parameter is the trace file name. The input file contains Pascal Program source and the trace file contains a listing of the tokens found in the input file. Sample command lines together with corresponding actions by program **pas** are shown below. Boldfaced type indicates data entered at the keyboard by the user.

```
$ pas
Enter the input file name: t01.pas

$ pas t01.pas

$ pas t01.pas t01.trc
```

Input File: The input file contains a micro program. File **t00.pas** is found in the class directory, **~tt/cs3613**, contains the source shown in Figure 1. Table 1 contains a listing of the Micro tokens.

Trace File: The trace file contains a listing of the tokens found in the input file. An example of the output is shown in Figure 2. The trace file name has the same prefix as the input file name and the suffix **.trc**.

```
program example(input,output);
      var x,y:integer;
      function gcd(a,b:integer):integer;
      begin{gcd}
            if b=0then gcd:=a else gcd:=(b,a mod b)
      end;{gcd}
begin{example}
      read(x,y);
      write(gcd(x,y))
end.
```

Figure 1. Input file **t00.pas**

```
LEXEME                               SPELLING
PROGRAM                              program
ID                                   example
LPAREN                               (
ID                                   input
COMMA                                ,
ID                                   output
RPAREN                               )
SEMICOLON                            ;
VAR                                  var
ID                                   x
COMMA                                ,
ID                                   y
COLON                                :
ID                                   integer
SEMICOLON                            ;
FUNCTION                             function
ID                                   gcd
LPAREN                               (
ID                                   a
COMMA                                ,
ID                                   b
COLON                                :
ID                                   integer
RPAREN                               )
COLON                                :
ID                                   integer
SEMICOLON                            ;
BEGAN                                begin
IF                                   if
ID                                   a
EQU                                  =
ID                                   b
THEN                                 then
ID                                   gcd
ASSIGN                               :=
ID                                   a
```

Figure 2. Output file **t00.trc**

```
LEXEME                                SPELLING
ELSE                                  else
ID                                    gcd
ASSIGN                                :=
ID                                    gcd
LPAREN                                (
ID                                    b
COMMA                                 ,
ID                                    a
MOD                                   mod
ID                                    b
RPAREN                                )
END                                   end
SEMICOLON                             ;
BEGAN                                 begin
ID                                    read
LPAREN                                (
ID                                    x
COMMA                                 ,
ID                                    y
RPAREN                                )
SEMICOLON                             ;
ID                                    write
LPAREN                                (
ID                                    gcd
LPAREN                                (
ID                                    x
COMMA                                 ,
ID                                    y
RPAREN                                )
RPAREN                                )
END                                   end
PERIOD                                .
```

Figure 2. Output file **t00.trc** (continued)

Lexeme	**Pattern**	**Lexeme**	**Pattern**
AND	and	EQU	=
ARRAY	array	NEQ	<>
BEGAN	begin	LES	<
DIV	div	LEQ	<=
DO	do	GRT	>
DOWNTO	downto	GEQ	>=
ELSE	else	PLUS	+
END	end	MINUS	-
FOR	for	STAR	*
FUNCTION	function	SLASH	/
IF	if	ASSIGN	:=
MOD	mod	LPAREN	(
NOT	not	RPAREN	)
OF	of	LSQBRACKET	[
OR	or	RSQBRACKET	]
PROCEDURE	procedure	COLON	:
PROGRAM	program	SEMICOLON	;
THEN	then	COMMA	,
TO	to	PERIOD	.
VAR	var	RANGE	..
WHILE	while		
ID	(*letter*|_)(*letter*|*digit*|_)*		
INTLIT	*digit*+		
REALIT	*digit*+\\.*digit*+(E(+|-)?*digit*+)?		
REALIT	*digit*+E(+|-)?*digit*+		
CHRLIT	\’[^’]\’		
CHRLIT	\’\’\’\’		

Table 1. Pascal Token Specification