User Guide

PL0 Compiler

Harold Joace Marcial
Ashton Ansag

How to Compile

To compile and run the compiler:

Note: Program assemblied with provided Makefile.

6 different commands

"make" just compile the program

"make lex" compile and run the lexer

"make parse" compile, run lexer, and run parser

"make parseL" compile, run lexer, and run parser with -l command to show lexemelist.txt

"make compile" compile, run lexer, run parser, and run code

"make compileL" compile, run lexer, run parser with -l command to show lexemelist.txt, and run code

Example Codes

Example 1: (begin-end statements)

```
var x, y, sum; /* this makes three variables to be used in the program
*/
begin
     x := 10;
     y:=32;
     sum := x + y; /* sum will have the vaule of 42*/
end.
```

Example 2: (if-then statements)

```
var global;      /* this is a global vaule to be used anywhere*/
procedure foo;
     var local;  /* this only exsists in this procedure */
     begin
          local := 2 * global; /* value is 42 until procedure ends*/
     end;
begin
     global := 21; /* value is 21 everywhere*/
     /* if <condition> then <statement> else <statement> */
     if global <> 42 then call foo else /* nothing */;
end.
```

Example 3: (do-while statements)

```
var x, squ;
procedure square;
begin
     squ:= x * x;
end;

begin
     x:= 1;
     while x <= 10 do
          begin
               call square;
               x := x + 1;
               end;
end.
```

Table of Reserved Words/Symbols

Reserved Words:

const
var
procedure
call
begin
end
if
then
else
while
do
read
write
odd

Special Symbols:

'+', '-', '*', '/', '(', ')', '=', ',' , '.', ' <', '>', ';' , ':' .

Identifiers:

identsym = letter (letter | digit)*

Numbers:

numbersym = (digit)*

Invisible Characters:

tab, white spaces, newline

Comments denoted by:

/* . . . */

SOME NOTES:

```
- := means becomes
      ex: 'x := (becomes) 100;'
- = is only used for constants.
      ex: const x = 3;
- <> means not equal
      ex: 'if x <> 100 then x := 100 ;'
- call is used to make a procedure call
      ex: 'call foo'
```