PL/0 User Manual

Raymond Cload

Christopher Beir

**Table of Contents**

1. How to compile and run the PL/0 compiler
2. How to use the compiler flags
3. Basics of the PL/0 language
4. EBNF of PL/0 language
5. How to compile and run run the PL/0 compiler

To compile and run this PL/0 compiler in a Linux environment, simply run the command:

sh CompileDriver.sh

This will create programs called 'scanner', 'parser', and 'vm' and take a PL/0 program from 'input.txt', compile, and run it.

The output of this program can be found in their respective output files ('cleanInput.txt', 'lemxemelist.txt', 'lexemetable.txt', 'mcode.txt', and 'stacktrace.txt').

1. How to use the compiler flags

Output can also be sent to the terminal window by compiling with the following flags after ‘sh CompileDriver.sh’.

-l : print a list of lexemes/tokens (scanner output) to the screen

-a : print the generated assembly code (parser/codegen output) to the screen

-v : print the virtual machine execution trace (virtual machine output) to the screen

1. Basics of the PL/0 language

Below is a commented program written in PL/0 meant to demonstrate the basics of the language:

/\* Any text enclosed like this is a comment, and is ignored by the compiler \*/

const j=1; /\* Constants must be declared in the const section of the procedure, this section is not required \*/

var x,y,z,v,w; /\* Variables must be declared after any constants \*/

procedure a; /\* Any nested procedures should be declared before the code block of the parent procedure \*/

var x,y,u,v;

procedure b;

var y,z,v;

procedure c;

var y,z;

begin /\* The code for a procedure should be enclosed by ‘begin’ and ‘end’ \*/

z:=j;

x:=y+z+w; /\* The language supports +,-,\*,/, and % (modulus) operations \*/

end;

begin

if j < 5 then /\* When using if-then statements, you do not add a semicolon to the end of any line except the last, unless your if statement includes ‘begin’ and ‘end’ \*/

y:=x+u+w

else

y:=x+u;

call c;

end;

begin

z:=0;

while z < 2 do /\* Similar to if-then, while statements also only have semicolons at the end \*/

z:=z+1;

u:=z+w;

call b;

end;

begin

x:=1; y:=2; z:=3; v:=4; w:=5;

x:=v+w;

write z;

call a;

end.

1. EBNF of PL/0 language

To give a more advanced look at the PL/0 language, the EBNF is included below:

program ::= block "." .

block ::= const-declaration var-declaration procedure-declaration statement.

constdeclaration ::= ["const" ident "=" number {"," ident "=" number} ";"].

var-declaration ::= [ "var "ident {"," ident} “;"].

procedure-declaration ::= { "procedure" ident ";" block ";" }

statement ::= [ ident ":=" expression

| "call" ident

| "begin" statement { ";" statement } "end"

| "if" condition "then" statement ["else" statement]

| "while" condition "do" statement

| "read" ident

| "write" ident

| e ] .

condition ::= "odd" expression

| expression rel-op expression.

rel-op ::= "="|“<>"|"<"|"<="|">"|">=“.

expression ::= [ "+"|"-"] term { ("+"|"-") term}.

term ::= factor {("\*"|"/") factor}.

factor ::= ident | number | "(" expression ")“.

number ::= digit {digit}.

ident ::= letter {letter | digit}.

digit ;;= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9“.

letter ::= "a" | "b" | … | "y" | "z" | "A" | "B" | ... | "Y" | "Z".

Based on Wirth’s definition for EBNF we have the following rule:

[ ] means an optional item.

{ } means repeat 0 or more times.

Terminal symbols are enclosed in quote marks.

A period is used to indicate the end of the definition of a syntactic class.