**PL/0 User Guide**

Gregory McHugh

At the very beginning of your PL/0 file will be your declaration of any constants, then declaration of any variables, and then declarations of any procedures. The name of a constant, variable, or procedure is an identifier, a series of alphanumeric characters, starting with an alphabetical character. An identifier can be no longer than 11 characters.

After any declarations, the programs main block is written, followed by a period. The main block is composed of the keyword “begin”, then the statements to be run, and ends with the keyword “end”.

Each statements must end with a semicolon unless it is the last statement in a block.

Constants

For a constant declaration, the constant is initialized in the same statement.

The keyword const followed by the constant’s name, the equal sign, then the constant’s value, followed by a semicolon ;.

For a constant called “a” with a value of 10:

const a = 10;

If there are multiple constant declarations, then they can exist sequential in one statement.

const a = 10, b = 12;

Variables

For a variable declaration, the keyword var followed by the variable’s name, and then the semicolon ;.

For a variable called “a”:

var a;

If there are multiple variable declarations, then they can exist sequential in one statement.

var a, b;

Procedures

For a constant declaration, the procedure’s definition immediately follows.

The keyword procedure followed by the procedure’s name, followed by a semicolon ;, and then the procedure’s definition. The procedure definition is composed as a block, beginning with the keyword begin and end with the keyword end followed by a semicolon.

For a procedure “example” that sets a variable “a” to the value of 3:

procedure example;

begin

a := 3;

end;

There can be multiple multiple procedure declarations, just create them one after the other as shown:

procedure example;

begin

a := 3;

end;

procedure another;

begin

b := 7;

end;

Using Procedures

A procedure may be used anywhere in a procedure or the program’s main code block by using the “call” keyword.

Example:

call proc;

Here, a procedure called “proc” is run and the body of that procedure is run as if it were in that call statement’s place.

Expressions

Expressions are composed of constants, variables, numbers, or operators. A number can no longer five digits.

Mathematical Operators: +, -, \*, /, :=

Those first four operators are recognizable as addition, subtraction, multiplication, and division. The last is the assignment operator, which assigns a value to a variable.

Here a variable “a” is assigned a value of 3.

a := 3;

Conditional Operators: <, >, <=, >=, =, odd

Those first five operators are recognizable as less than, greater than, less than or equal to, greater than or equal to, and equal to. The last operator returns 1 if the expression to its right has an odd value, and 0 if it’s even.

odd a \* b

In the above example, the result is 1 if a \* b is an odd number, and 0 if it is even.

Conditional Statements

PL/0 includes conditional statement structures for control flow. A simple if statement is composed of the keyword “if”, a conditional expression, the keyword “then”, and then the the statement to be run (which may be a block).

A basic example is a simple “if then” statement:

if input > 5 then

ans := 5;

If an “if then” statement is intended to running multiple statements instead of one, then a block must be formed by surrounding those statements with the keywords “begin” and “end”.

In an “if then”, the user may optionally use the key word “else” to create an “if then else” statement.

if input > 5 then

ans := 5

else

ans := 0;

The user may even incorporate an “if else if else” structure.

if input > 5 then

ans := 5

else if input > 4 then

ans := 4

else if input > 3 then

ans := 3

else

ans := 0;

Input & Output

Input from a user can saved to a variable using the “read” keyword. When the program runs a read statement, the program will pause and wait for the user to type a value and press enter.

read a;

A value or a constant can be output to the screen using the “write” keyword.

write a;

Statement Blocks

A block is composed of the keyword “begin”, then the statements to be run, and ends with the keyword “end”. Each statement must end with a semicolon unless it is the last statement in a block.

begin

a := 3;

b := 7

end;

Take note that a statement can be an empty string.

Loops

PL/0 provides support for a loop structure in the form of the “while do” loop. Such a loop is composed of the keyword “while”, then a conditional expression, the keyword “do”, followed by a statement to be run (which may be a block).

An example:

while i > 1 do

begin

n \* i;

i := i – 1

end;

Using the compiler

Compile the compiler.c file.

$gcc compiler.c

Then run the resulting executable. On default, it will be named "a.out"

(unless you used "-o" to give it a different name, but for these instructions I'll assume you didn't).

$./a.out

The executable will be expecting the PL/0 code to be in a file named "input.txt" in the same directory.

All printed output will be printed out to a file "output.txt" in the same directory as the executable.

The scanner and lexical analyzer prints to "lexerOutput.txt".

The parser and code generator prints to "parserOutput.txt".

The virtual machine prints to "vmOutput.txt".

Included in this zip file is a sample program in a "input.txt" file and the resulting "output.txt" file.

Also included is a list of 25 errors that can be reported named "error-list.txt"

and a sample code with resulting output for each possible error.

These sample input and output files for these errors are all in the "errors" directory.