User’s guide to PL/0

Josh Ackerman

Scott Mosher

Variables

Just like with all machines the PL/0 machine relies heavily on variables. There are 3 kinds of variables standard which are declared variable “var x, y;”, constant “const a=0, z=8;”, and Procedure “procedure P”. When declaring variables and constants multiple can be declared at once so long as they are separated by a comma (,) the declaration must always end in a (;) as should just about all the code developed for PL0. For Constants, the value must also be assigned using the operator “=” once a constant is declared it cannot be changed. Attempts to change a constant will result in an error. Variable (var) can be declared any time though it can be used only after it’s declaration, so if you try to use before its declared an error will be thrown. Both const and var can be declared in procedures, once the procedure is over the var and const will both be dropped from the registry and be unobtainable. Procedure is a variable only insofar as it is called by its name. A procedure is a location for the code to jump to it can be reached by using the code “call P” the code will then go to the location where P was declared and run the code until it finds and reaches an end for which there was no begin other than the one that started the procedure. If the variable for a procedure is used in anything other than a call function an error will be thrown. The code will initially skip by code covered by a procedure.

…………………………………………………………………………………..

Math and Assignment

Math and declarations are relatively similar to other programing languages. In order to use a variable first, you must make sure it is declared with a value to do this you must always use the assignment symbol “=:” and end with a semicolon (;) for instance “x=: 30;” now x can be used as a variable for instance “y=: x+z;” when declaring variable, it the available functions are addition (+) multiplication (\*) division (/) and subtraction (-). If you wish to have a multiple part equation such as “x=x+y/z;” you can, it will activate in pemdas order. (the standard for math of all types) if you wish to violate pemdas you can use parenthesis if you want x+y first for instance “x=(x+y)/z;”. Any value can be negated by putting a “- “sign in front of it so long as it follows the standard rules.

………………………………………………………………………………………………………………………………………………………………

If, Else, While and conditions

If you want a condition to only activate under certain conditions use an if statement the conditions available are, < less than, <= less than or equal, <> not equal, = equal, > greater than, >=, and odd (ex if odd x then x=:x+1;) simply put the variable that fits the description you want on the left and the second variable on the right, with the exclusion of odd which has only 1 variable which goes on the write. If these instructions are not met due to the use of a procedure, inclusion of too few or too many variables an error will be thrown. These conditions work the same way in the while loop. For an if statement once the condition is put down it must be followed by a then, lack of a then will create an error, which will give an instruction of what to do. This instruction only lasts one line with the exception of a begin statement which lasts until an end statement is found. If there is a case that should occur only if the if statement is not met you can put an else statement (though an else statement is not required) after the action of the if but before any further code, the else will follow the same pattern as the if and again only activate if the condition for if was not met. While loop follows a similar pattern with a few exceptions. To use a while you must first put in a condition followed by a “do”, lack of “do” will lead to an error, detailing the action to occur if it’s begin it will go until it finds an end otherwise it will just go one line. Once the while loop has gone to its limit it will check the condition again if the condition is met the while loop will move on if the condition is not met it will continuously compete the action(s) until the condition is met. For example, “while x>30 do x=:x-2;” Infinite loops do not receive errors so be careful.

Read and Write

If you want to be able to print something to the screen use the write command for example “write x;”, attempting to write a procedure will end in error, to input a value use read for example “read x;”, attempting to read a procedure or const will end an error.

Begin end and “.”

begin essentially starts the program or tells the program that the next set of instructions go together and shouldn’t be broken up until an end statement is read at which point it should return to whence it came. In order to completely end the program, use an “end.” Without one the program will not end properly.

Compiling

In order to compile your code first, put it into a file named input.txt next use the instructions

gcc -o d pl0compiler2.c (where d is any name) afterwards use (with d being you chosen name)

./d -l -a -v

in order to run everything, -l shows scanner, -a parser and -v vm. Make sure to check out input, finaloutput, output, and lexemelist text files, if you want that sort of information. Any information you asked to input or display using read and write will be output to the screen, you can do whatever you wish to it at that point.