PL/0 Compiler User's Guide

Authored by Dallas Seroski, David Allen

Version 1.0

Released: April 14th, 2017

1 .0 How to Compile and Run the PL/0 Compiler

1.1 Setting Up the Platform to Run the Compiler

1. Download the source code folder for the compiler.
2. If the system doesn't have GCC and GNU installed, install them to compile the source code in the compiler.
3. Read the readme file on that comes with the compiler source code to understand how the compiler works

1.2 Compiling the source code for the compiler

1. Open the command prompt and navigate to the folder where the source files are.
2. Type "`gcc -o ./compiler ./src/compiler.c ./src/errorHandler.c ./src/icg.c ./src/P-machine.c ./src/parser.c ./src/scanner.c`"
3. This compiles the source files and makes an executable compiler file.
4. Test to see if a compiler file was actually made by typing "`./compiler`".
5. The output should tell you there are missing arguments and show the expected input along with what each input does.

2.0 How to Use the PL/0 Compiler Once it is Running

2.1 Running the Compiler with a PL/0 file

The compiler needs at least 1 argument to run correctly and that is the .pl0 file. To do this you must execute the compiler with the file.
For example: "`./compiler ./GoodPL0files/test.pl0`" should print out "`No errors, program is syntactically correct.`" To the command prompt.

2.2 Running the Compiler with a PL/0 file and arguments

The compiler also takes in 3 commands the user can pick to output.

1. "-l" prints out the list of lexemes tokens that are associated with the PL/0 file.
2. "-a" prints out the object code associated with the PL/0 file.
3. "-v" prints out the Virtual Machine execution trace.

These commands can be run in any order and any combination.
For example: "`./compiler -a -l ./GoodPL0files/test.pl0`" should print out:

3.0 How to Use the PL/0 Language

PL/0 works with constants, variables, and procedures. The grammar for this language is shown below. Generally speaking, there are declarations for constants, variables, and procedures followed by the block of code.

EBNF of PL/0:

```
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" expression
                | 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.

3.1 Comments

Users can declare comments using the “/*” character to sequence and “*/” to end the comment

Example:

```
      begin /* proc c */
        z:=1; /* z is local */
        x:= y + z + w; /* x_a = y_c + z_c + w_base = (0 + 1 + 5) = 6 */
        write x; /* should print 6 */
      end;
```

3.2 Data Types

There are 3 data types that are supposed by the PL/0 language: Constants, integers, and procedures. Each of these data types must have an identifier.

3.2.1 Constants

Constants are integer types and may only be defined once in the program. You can define more than one constant at a time. Once an constant is defined in the program you cannot change their value.

Example:

```
const foo = 3, bar = 2;
```

3.2.2 Integers

Integers are like constants however they are changeable once they are defined.

Example:

```
var foo, bar;
begin
    foo:= 4;
end
```

This changes foo to 4.

3.2.3 Procedures

Procedures are the functions or methods in a program. Procedures can be nested in one another also.

Example:

```
var x,y,z,v,w;
procedure a;
  var x,y,u,v;
  procedure b;
    var y,z,v;
    procedure c;
      var y,z;
      begin /* proc c */
        z:=1; /* z is local */
        x:= y + z + w; /* x_a = y_c + z_c + w_base = (0 + 1 + 5) = 6 */
        write x; /* should print 6 */
      end;
    begin /* proc b */
      y:= x + u + w; /* y_b = x_a + u_a + w_base (0 + 7 + 5) = 12 */
      write y; /* should write 12 */
      call c
    end;
  begin /* proc a */
    z:=2; /* sets z_base to 2 */
    u:= z + w; /* sets u_a to 2 + w_base (5) = 7 */
    write u; /* should write 7 */
    call b
  end;
begin /* main */
  x:=1; y:=2; z:=3; v:=4; w:=5;
  x:= v + w; /* x_0 should be 9 now */
  write x; /* should write number 9 */
  call a;
  write w;
end.
```

3.3 Expressions

Expressions in the PL/0 language are basic mathematical expressions. They consist of multiple symbols including: +, -, *, /, (, and).

3.4 Input and Output

The PL/0 machine works with input and output using the read and write commands.

Example:

```
var x, w;
begin
    x:= 4;
    read w;
    if w > x then
        w:= w + 1
    else
        w:= x;
    write w;
end.
```

Read takes w in as input from the command prompt. Write sends the value of w to the command prompt.

3.5 Assignment

To assign a value to a variable the PL/0 language accepts ":=" to assign an expression to a variable. Notice on constants declarations the PL/0 language uses a regular "=".

Example:

```
        w:= w + 1
```

3.6 Conditional Statements

The PL/0 machine supports if-then else and while do statements.

3.6.1 If-Then Else

If-then else statements can be call with the following syntax

Example:

```
    if w > x then
        w:= w + 1
    else
        w:= x;
    write w;
```

3.6.2 While do

While statements work similar to If-then statements and can be used with the following syntax.

Example:

```
while x > 3 do x := x - 1;
```

3.7 Call

Call is used for procedures and is the only way to call the function.

Example:

```
  call foo;
```

3.8 Recursion

The PL/0 compiler supports recursion. Here is a basic factorial example of the program.

Example:

```
var f, n;
procedure fact;
var ans1;
begin
  ans1:=n;
  n:= n-1;
  if n = 0 then f := 1;
  if n > 0 then call fact;
  f:=f*ans1;
end;

begin
  n:=3;
  call fact;
  write f;
end.
```