Atto (Plus, Minus) Interpreter User Guide 1.0

It's a simple interpreter designed to handle basic control instructions, mathematical, and logical operations. It can be compiled with the Free Pascal compiler and is aimed at executing fundamental scripting tasks like calculating the Fibonacci or Collatz sequences.

Compilation and Execution

Use the following command to compile the program: fpc atto.pas (or attoPlus or attoMinus.pas)

Execution: Run the program by specifying input and output files, for example:

```
atto.exe < FIBONACCI > FIBONACCI.OUT
```

The generated output file, such as FIBONACCI.OUT, will contain the results of program execution.

Basic Rules

Variables

The interpreter uses built-in variables. They are identified by the English ABC letters (A..Z).
The first 16 (A..Q) can store 32-bit signed integer values. It functions as a generator of random numbers with the identifier R and the variables with the identifier S..Z store the code of 8-bit ASCII characters. The latter are used together with the PRN command to print the character corresponding to the code on the screen or in the output file.

Spaces: In Atto (AttoPlus, AttoMinus) scripts, every element (e.g., number, keyword, variable, operator) must be separated by a space.

```
Correct..: B = B + 12
Incorrect: B=B+12
```

Keywords and Their Functions

The interpreter recognizes four keywords, each with its own function:

IF - Conditional branching
After IF, a logical condition must be provided. If the condition is true, the instruction following IF will execute. IF can be used with JMP, PRN, or assignment instructions.

```
Example: IF A < 10 JMP .LOOP
```

This means that if the value of variable A is less than 10, the interpreter jumps to the label .LOOP.

JMP - Jump to label
In addition to conditional usage with IF, JMP can be used on its own to modify the execution sequence by jumping to a specified label.

```
Example: JMP .END
```

This command jumps to the .END label in the program.

PRN - Print
The PRN instruction outputs the value of the specified variable. For ASCII characters (S-Z), it outputs the character, while for numeric variables (A-R), it outputs the numeric value.

```
Example PRN A S B
```

This prints the current value of the variable A (space char) B.

RET - Return from subroutine
The RET keyword is used to return control to the last location specified by a JMP statement. Primarily, this is used to exit subroutines.

Example: RET

If a subroutine is called (e.g., JMP .SUBROUTINE), RET brings execution back to where the JMP was made.

Using Labels

Labels are names for specific locations in the program to which jumps or subroutine calls can link. Each label begins with a . character, allowing the interpreter to navigate within the code.

Example:

```
.LOOP
A +
PRN A
IF A < 10 JMP .LOOP
```

Here, .LOOP serve as label marking logical section of the program, enabling jumps and conditional branches.

Mathematical and Logical Operations

Math Ops: Atto supports following ops: +, -, *,/, % (modulus).
Log Ops.: Conditions can use <, >, # (not equal), = operators for evaluation.

The interpreter is very simple, so it cannot work with complex mathematical or logical expressions. The language doesn't even recognize parentheses. Parentheses are completely unknown in the ATTO language.
What are complex mathematical/logical expressions for the interpreter? Those that contain more than one mathematical or logical operator in one expression. The interpreter cannot handle such things, so they must first be broken down into simple expressions.

Example of compound expressions:

```
B = B + C * 2
```

This should be broken down, something like this:

```
C = C * 2
B = B + C
```

The same for a logical expression:

```
IF B < D + 1
```

This is not good because there is more than one operator in the expression. Simplification:

```
D = D + 1
IF B < D
```

or

```
D+
IF B < D JMP .LOOP
```

Syntax Errors

The interpreter detects and reports syntax errors, such as poorly formatted expressions or missing spaces. This error handling facilitates development and debugging.

A debug/Trace mode can be activated using the **TRC** instruction, which displays the current values of variables and program state. This can be useful when testing Your Atto scripts.