Assignment Part 2 – Algol 68

# Program Testing Procedure

Run program, observe console output. Every 3rd line should be "Fizz", every 5th line "Buzz", every 15th line "Fizz buzz". The other lines should contain the line number.

Also make sure the output formatting is correct – Algol, like Fortran, defaults to using interesting formatting.

# Weekly Question

# Reflection

Algol’s adherence to the regularity principle with respect to program structure improves readability. Since every part of the program consists of a hierarchical block structure that relates directly to scope, it is very easy to identify name binding.  
A corollary of this regularity is that the language also adheres to the structured program principle (at least for this small program, that does not need forward declarations). It is very easy to follow the control flow of the program (aided by the block nesting), increasing readability.

Algol’s disallowance of implicit variable declaration and usage of reserved keywords adheres to the defence in depth principle, improving reliability. While I was figuring out the syntax of the language, I made various keyword and name related mistakes, all of which were immediately detected by the compiler – once the program compiled successfully, I was confident the program executed as desired.

Some aspects of Algol’s syntax somewhat violate the syntactic consistency principle, reducing readability and writability (at least for a beginner).  
For example, the modulo operator is %\* - with the \* operator performing multiplication, and the % operator performing integer division. This makes little sense, as each of these is distinct operations.  
Equality checking is performed with the = operator, while assignment is performed with :=. Again, these are distinct operations sharing very similar syntax, and are likely to trip up those used to = for assignment, for example from Fortran or C.

Algol’s I/O formatting violates the simplicity and regularity principles, reducing readability and writability. The formatting behaviour of print() changes depending on the type of the argument – numbers are by default aligned, but strings are not. Furthermore, it is, in my opinion, too difficult to change these default behaviours. For example, in order to print a number n without any formatting, one can use print(whole(n, 0)). But in order to also include a trailing newline, one must use print((whole(n, 0), newline)) (note the extra parentheses), or printf(($%dl$, n)) (using the “format-text” syntax, which is quite complex and unintuitive).