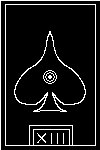
**SPADE**

**Version: 3.0.6**

**The User Manual**



**Introduction**

SPADE (Simple Parsing Algorithm Developed using C++ Expressions) is an esoteric1 programming language which was first conceived for a grade 12 project by its creators, Kiran Bodipati (@Kilobyte?) and Subienay Ganesh (@swampie27782). It is a stack-based language, though at first glance it may look like BASIC with elements of C. It is unique in having essentially the same functions as BASIC, yet being far more powerful. The program is powerful, because using the goto statements, normally considered as bad programming, along with clever use of the stack nature of variables, SPADE can solve a large compendium of complicated tasks. As of now, it is yet to be proved to be Turing-complete.2

HELLO WORLD!

printl “Hello World!.

**It’s as simple as that.**

**THE SYNTAX AND COMMON FUNCTIONS**

All SPADE statements have to end with the fullstop (.) and every statement has to begin in a new line from the beginning.

Never disrespect the full-stop. Indentation using tabs or spaces is not allowed. The simplest SPADE statement is the full-stop.

.

This statement can be used to make code more readable. There can be no input in SPADE.

1. **print and printl**

These functions are used to output any expression or constant.

Printl adds a newline to the end of the output.

**Syntax:**

print[l] [“constants”]<space>[(expressions in brackets)]<space>[“other constants etc...”].

1. **var**

The var keyword is used to declare variables.

Multiple variables can be declared in one line by adding a space between each variable. When a variable is declared, it is added to the top of a variable stack, and there is no restriction on having multiple variables of the same name, which results in some pretty clever programming tricks (See Advanced topics).

**Syntax:**

var name1 [name2…].

1. **assignment(:)**

Assignment of variables is done using the ‘:’ symbol. Variables can be assigned to be literals or expression values.

The three types of literals are:

1. String (Enclosed in “ ”)

2. Floating point numbers (Enclosed in ‘ ’)

3. Integers (Not enclosed).

Expressions are by default Floating point numbers.

**Syntax:**

name1 : 3.

name2 : ‘3.456’.

name3: “Hello world!”.

1. **escape sequences**

To prevent clashing of keywords with output and data, the \ character is used to differentiate between keywords and literals. Just add a slash before a keyword (like \for,

\:, \var, \. , \\ etc.) to convert it to a literal. Exceptions are print and printl, in which you have to place the slash before r (p\rintl).

1. **goto**

Goto is a very important function in SPADE. Goto is used to ‘teleport’ to any line in the program file. It assumes that the first line of code is number 1, the second 2, the third 3 and so on.

**Syntax:**

goto [line number].

1. **if-endif**

The if block is used to execute commands based on expression values. If the given expression value is 0, it ignores every statement between the two keywords. Otherwise, it executes the block.

**Syntax:**

if (Expression).

[Statements here].

endif.

1. **for condition**

The for condition is used to perform loops. It consists of three dependents, the loop variable, the limit and the condition. The for loop iterates until the condition given is unable to be satisfied.

**Syntax:**

for [variable or literal] step [variable or literal] till [variable or literal].

[Statements].

endfor.

1. **del**

The del keyword is used to delete a given variable. It deletes the topmost instance of the variable from the variable stack.

**Syntax**:

del [variable].

And that’s all the functions! Simple, right?

1. **Operators and Precedence**

The order of precedence of operators is

1. Exponent (^)
2. Multiplication (\*), Division (/), Modulus (%)
3. Binary Addition(+) and subtraction(-)
4. Greater than (>) and less than(<)
5. Relational Equal to(=), not equal to(~)
6. Logical AND(&), Logical OR(|)

**A FEW EXAMPLE PROGRAMS**

1. **Factorial**

Calculates the factorial of 10.

var a b.

a : 1.

b : 1.

print "The factorial of 10 is ".

for a step 1 till 10.

b : (b\*a).

endfor.

print b "\.".

**Output:**

The factorial of 10 is 3628800.

1. **Digital Root**

Find the digital root of a number – that is, the single-digit number you get when you add all the digits of a given number continuously.

var a root temp.

.

This program p\rints the digital root of a number.

.

a : '47358'.

temp : (a).

root : 0.

.

for a step 0 till 0.

.

root : (root+(a%10)).

a : ((a-(a%10))/10).

.

endfor.

.

if (root>9).

a : (root).

root : 0.

goto 7.

endif.

.

print "The digital root of " temp " is " root ".".

**Output**:

The digital root of 47358 is 9.

**~A note on readability~**

Comments can be written in SPADE – in fact, it’s very easy! Just type it down like it’s just a text file! However, end it with a fullstop and make sure to add escape sequences on any keywords.

**ADVANCED TOPICS**

1. **Implementing while loops**

While loops do not have a function in SPADE. Instead, you can use a for loop to perform the same functionality. Making the step count as 0 effectively converts the loop into while.

Therefore,

while (a != 0)

can be converted to

for a step 0 till 0.

1. **If and else if**

You may have noticed that there is no functionality for an else if condition in SPADE. This is where goto comes in. You can use goto to move to the end of a sequence of if blocks while writing your code.

Make sure to write down the appropriate line number for goto only after writing all the code you need!

1. **Using the variable stack**

The variable stack may seem useless at first, but is actually very powerful, because it allows you call a piece of code over and over again, while performing different functions each time. A simulation of recursion is very possible using this variable stack, as well as implementation of arbitrarily large numbers.

1. **The ~ symbol**

The ~ symbol is the ‘not’ symbol for using expressions, but it also has a special use – when placed in front of a string variable while printing, it ignores all escape sequence characters marked with \ , and prints even the slashes along with the characters.

QUIRKS

**Quines!**

Quines are possible in SPADE! A quine is a program which, on running, prints itself. Obviously, this requires a bit of tricky programming, but it is very possible in SPADE, and can be done using 3 lines.

var a.

a : "\\var a\.\\na \\: \\"\" ~a \"\\"\.\\np\\rint \\"\" a\.".

print "\var a.\na \: \"" ~a "\".\np\rint \"" a.

This program prints itself on running. How cool!

**The “esotericity” of the language**

This language was designed to be esoteric, so obviously, it must have some esoteric elements. The obvious features are the absence of important coding frameworks such as classes, functions, while loops, or else if statements, unconventional writing structure and lack of input. One less obvious feature is that the full-stop operator is not even needed for the language, since every line is required to start for a new line anyway! These, coupled with various other complications involving the variable stack and goto statements, make SPADE a proud esoteric language.

**Go ahead and test its limits!**