CS148

Ababon, Gian  
Chu, Kevin  
Lim, Erwin  
Pineda, Kevin  
Sabarre, Joseph

FTS Language Documentation

**Basic Syntax**

In FTS, every statement must be ended with a semi-colon (;). One-line comments may be made by using two slashes (//). Statements must also follow the rules of the operators and control statements involved. Operators in FTS (unless explicitly mentioned otherwise) follow infix notation.

**Variables**

Every variable in FTS starts with the $ symbol. Scalars and arrays are possible. Scalars are variables which hold strings, integers, and floating-point numbers. Arrays are ordered collections of scalar values which are retrieved using numeric index where 0 is the first index.

For example:

$myInt = 1;

$myFloat = 2.0;

$myString = “Hello”;

Explicit data types in FTS are optional. These data types would be int, float, and string.

For example:

int $myInt;

float $myFloat;

**Operators**

The assignment operator is denoted by the = operator. This operator would copy the value on the right side to the variable on the left side.

For example:

$x = $y;

Binary arithmetic operators include addition (+), subtraction (-), multiplication (\*), division (/) and modulus (%). Unary arithmetic operators include the prefix and postfix autoincrement (++) and autodecrement (--). The prefix operators modify their argument before it is evaluated while postfix operators modify it after.

There are comparison operators. These would be less than (<), greater than (>), less than or equal (<=), greater than or equal (>=), equal to (==), and not equal to (!=).

There are logical operators. These would be AND (&&), OR (||), and NOT (!).

There are also bitwise operators. There would be the bitwise AND (&), bitwise OR (|), bitwise XOR (^), one’s complement (~), and the shift operators (<< for left shift and >> for right shift).

There is also the ternary operator. Its structure follows (condition) ? (return value for true) : (return value for false).

**Control Statements**

Plain blocks are blocks surrounded by curly braces ({}). These are not preceded by anything, not even a condition. These blocks will execute once. These may be used for temporary variables whose scope does not extend beyond the block.

There is also the “if” control statement. The keyword “if” must be used followed by parentheses. There must be a condition within the parentheses. The condition must evaluate true for the block (or single statement) following the condition to be executed. A false condition will result in the block not being executed.

There is also the “jump” control statement. It functions like the “switch” statement in programming languages like C. The “case” keyword is also replaced with “value”. A colon follows the value of “value”. Just like in C, value statements will cascade if not terminated with a keyword. This keyword in FTS is “stop”.

There is also the “amid” control statement. It functions like a “while” statement in C. There is a condition which must evaluate true for the block (or single statement) that follows for it to be executed. As long as this condition remains true, the loop will continue. It is therefore necessary to find a way to make the condition false somewhere within the loop. The “stop” keyword may also be used within the loop to force an exit.