SPL language overview

Overview

SPL is a small lightweight interpreted programming language which is designed to operate on multiple possibly infinite input streams and create out of it a single output stream. To minimise the amount of code required to perform a task the concept of streams has been abstracted away from the language to allow the user to focus on single elements of the stream at a time and have the interpreter take care of the stream of data.

Types

What types do we have and why

Syntax

Syntax	Description
var = [value];	Assigns [value] to the variable named var. Can be placed at the start of the program file to assign a golbal varaible or within the process block to assign a local variable. For more information see the variables section
process{...}	The process block contains the code to process the input stream. This block will be executed K times where K is the number of elements in the input stream
stream[N];	Has the value of the head of the input stream N.
inject([value]);	Injects the value [value] into the output stream. This is what is used to build up the result of the calculation

Variables

Variable names can include any alpha-numeric character or an underscore as long as the variable names do not start with a number and is not one of the reserved keywords. The reserved keywords are *if*, *else*, *while*, *TRUE*, *FALSE*, *process* and *inject*.

Scope

There are 2 types of scope within SPL (global and local). Global variables are declared at the top of the program file and will have persistence throughout the entire run of the program. Local variables are variables which are declared within the process block. Local variables will only have meaning for the remaining portion of the process block and once that part of the stream has been declared the values are thrown away. Local variables should be used for temporary calculations within the block while global variables should only be used when it is required to store some state in between the different elements of the input stream (for example counting how much of the stream has been consumed so far).

Running a spl program

```
splinterpreter myProgram.spl < input
```

Common errors

Doing things wrong.