ZORML 2.2 - (Dec 2024)

Basic Syntax

ZORML is a stack-based programming language where each command is a single character. Commands manipulate the **stack**, execute functions, or interact with the user. Here's a breakdown of available commands:

Core Commands

Symbol	Description	Example
`>`	Pushes the next character onto the stack.	`>A^` → Stack: `A` → Prints `A`.
`<`	Prompts the user for input and pushes the entered value onto the stack.	`<^` → Input: `5` → Prints `5`.
`^`	Prints the entire stack as a single string.	`>A>B^` → Prints: `AB`.
`!`	Prints the value at a specific stack index.	`>1>2!0` → Prints: `1`.
`#`	Clears the entire stack.	`>A#^` → Prints nothing.
`[ ]`	Adds a string or long sequence of characters to the stack.	`[hello]^` → Prints: `hello`.
`{ }`	Defines a function to be executed later.	`{>A^}` → No output yet.
`~`	Executes the most recently defined function.	`{>A^}~` → Prints: `A`.
`;`	Stops the interpreter from processing further commands.	`>1>2;>3^` → Stops at `;`.

Mathematical Operations

Symbol	Description	Example
`+`	Adds the top two items of the stack (if they are numbers) or concatenates them as strings.	`>1>2+^` → Prints: `3`.
`-`	Subtracts the second item from the top item (if they are numbers).	`>5>3-^` → Prints: `2`.
`/`	Divides the second item by the top item (if they are numbers).	`>6>2/^` → Prints: `3`.
`*`	Multiplies the top two items of the stack (if they are numbers).	`>3>4*^` → Prints: `12`.

Conditional Logic

Symbol	Description	Example
`?`	Starts an **if condition** that checks if two stack indices are equal.	`>1>1?0=1[>A^]` → Prints: `A`.
`[ ]`	Encloses commands to be executed if the condition is true.	`?0=1[>B^]` executes if true.

Condition Syntax:

- `?index1=index2[commands]`: Executes commands if values at `index1` and `index2` are equal.

Examples

Print "hello world"
[hello world]^;

Add Two Numbers
>3>5+^; // Output: 8

If-Else Behavior
>5>5?0=1[>A^];>B^; // Output: A (doesn't print B because it stops at ;)

Define and Execute a Function
{>Hello^}~; // Output: Hello

Stop Execution
>1>2+^;>3^; // Output: 3 (stops at `;`)

Note:

1. The **stack** grows as you add elements and is cleared when you use `#`.
2. If a command uses indices, they start at `0` (like arrays in most programming languages).
3. Commands are case-sensitive.