Forth Golfscript Interpreter

Golfscript

Golforth

Code Golf

- shortest possible source code that implements an algorithm
- solving problems (holes) in as few keystrokes as possible

Golforth

Code Golf

- shortest possible source code that implements an algorithm
- solving problems (holes) in as few keystrokes as possible

Golfscript

- stack oriented, variables exist
- single symbols represent high level operations
- strong typed
- heavy use of operator overloading and type coercion

Golforth

Golfscript Types

- Integer: `1 2`
- Arrays: `[1 2 3] [3]`
- Strings: `"one two three"`
- Blocks: `{1+}`

Golforth

Golfscript Types

- Integer: `1 2`
- Arrays: `[1 2 3] [3]`
- Strings: `"one two three"`
- Blocks: `{1+}`

Golfscript Operator Example

- `12 3 * -> 36`
- `[50 51 52]' '* -> "50 51 52"`
- `[1 2 3]{1+}/ -> 2 3 4`
- `{.@\%.}do; ( n1 n2 -- gcd )`

Forth Implementation

Golforth

Typesystem

- Values as scalar references on stack
- Anonymous functions
  - `: anon_int { u -- typext }`
    `:noname u POSTPONE LITERAL POSTPONE typeno_int POSTPONE ; ;`

```
12 anon_int s" foo" anon_str
```

↓

```
2 elements on stack (12 and 'foo')
```

Golforth

Parser

- translates golfscript to forth execution tokens
- based on regular expression of reference implementation
- Responsible for:
  - infer initial type from syntax
  - symbol table for variable tracking
  - note that every value can be a variable!

```
12 'foo':x
```

↓

(creating x in symbol table)

↓

```
12 anon_int s" foo" anon_str dup x !
```

Golforth

Arrays

- Construction similar to postscript.
- **[** marks stack size, **]** collects back to marked size.
- Mark moves when stack becomes smaller:

```
1 2 [\] -> [2 1]
```

Golforth

Type Coercion and Overloading

- Typeorder for Coercion
- Coercion according to highest order type
- Heavy operator overloading results in wide range of functionality

Golforth

Implementation Status in Percentage

Complex Overloaded [1]	
!	100
=	88
~	100
$	0
+	100
-	75
*	0
/	0
%	25
,	75
?	75
(	75
)	75
|	0
&	0
^	0
<	0
>	0
and or xor	0
zip	0

Stack Operators	
@	100
\	100
;	100
.	100

Loops & Conditionals	
until	100
while	100
do	100
if	100

Math	
abs	100
base	100
rand	0

Output	
‘	0
print	100
puts	100
p	0

[1] Introspection based features for Blocks have not been implemented.

Golforth

Cutbacks

- Error Handling differs
- Introspective Block operations not implemnted

Golforth

Usage of Idiomatic Forth

- Stack paradigma mapped to typed language
- Wordlists for variable tracking
- Macros & anonym functions for
  - language implementation
  - operator implementation