Forth Golfscript Interpreter

Golfscript

Code Golf

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

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

Golfscript Types

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

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)

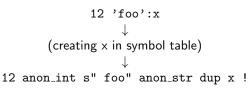
Forth Implementation

Typesystem

- Values as scalar references on stack
- Anonymous functions

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!



Arrays

- Construction similar to postscript.
- ▶ [marks stack size,] collects back to marked size.
- ▶ Mark moves when stack becomes smaller:
 - 1 2 [\] -> [2 1]

Blocks

- Stored as execution tokens
- ▶ Operations: $2\{1+\}+ \rightarrow \{2\ 1+\}$ implemented as function composition
- Execution via execute

Conditionals and Loops

- ▶ $5\{1-..\}$ do \rightarrow 4 3 2 1 0 0
- ▶ $5\{.\}\{1-.\}$ while $\rightarrow 432100$
- ▶ $5\{.\}\{1-.\}$ until $\to 5$
- implemented as words which consume code blocks

Type Coercion and Overloading

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

Hier koennten die demos beginnen:

Operator in shell zeigen und evtl entsprechende implementierung

```
Golforth
               100
               100
               100
               0
               100
               75 not for blocks
               0
               0
               25
               100
               100
               88
               75
               100
               75
               75
               75
and or xor
               0
print p n puts
               0
               100
do
while until
               0
if
```

abs zip

base

Cutbacks

- Error Handling differs
- Probably not all operators implemented
- Block operations not completly implemnted

Usage of Idiomatic Forth

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