Programação Orientada a Pilhas

Fundamentos

Prof. Edson Alves

Campus UnB Gama

Sumário

Visão geral

- A programação orientada a pilhas é um paradigma de programação que utiliza uma ou mais pilhas para a manipulação de dados e passagem de parâmetros
- O modelo de computação é o das máquinas de Turing
- A maioria das linguagens que suportam esse paradigma usam a notação pós-fixada para suas operações, isto é, os argumentos são informados antes da operação
- Como as operações manipulam a pilha, adicionando e removendo dados, é comum manter um registro das modificações feitas na pilha, denominado **diagrama de efeitos na pilha** (*stack effects diagram*)
- Expressões e programas podem ser interpretados de maneira simples e direta
- Exemplos de linguagens orientadas a pilhas: Forth, PostScript, Bibtex e Uiua

Forth

- Forth é uma linguagem baseada em pilhas, desenvolvida por Charles H. Moore nos anos 70
- Segundo o próprio autor, ela foi desenvolvida enquanto ele trabalhava em um IBM 1130, um computador de terceira geração
- Ele considerou que os resultados que tinha até momento eram tão bons que ele considerou que estava desenvolvendo uma "*fourth-generation computer language*"
- Daí surgiu a ideia de nomeá-la FOURTH
- Porém o IBM 1130 permita identificadores com, no máximo, cinco caracteres
- Por este motivo ele deu à linguagem o nome FORTH
- Forth foi utilizada pela NASA no desenvolvimento de aplicações para missões espaciais

GForth

- GForth é uma implementação *open source* da linguagem
- Em Linux, ele pode ser instalado com o comando

```
$ sudo apt install gforth
```

- Para checar se a instalação foi bem sucedida, rode o comando

```
$ gforth --version
```

- O GForth tem dois modos de operação: o interativo (REPL) e o modo interpretador, onde ele lê e executa as instruções contidas em um *script* Forth
- Para entrar no modo REPL, basta invocar o interpretador com o comando abaixo

```
$ gforth
```

- Para encerrar a sessão, utilize a palavra **BYE**

Interpretador GForth

- Para utilizar o GForth no modo interpretador, deve se escrever os comandos Forth em um arquivo de texto (*script*)
- A extensão adotada para *scripts* Forth é `.fs`
- Para interpretar um *script* Forth basta invocar o GForth, passando como argumento o nome do *script*

```
$ gforth script.fs
```

- O *script* abaixo implementa o tradicional "*Hello World!*" em Forth:

```
1 \ A palavra CR imprime uma quebra de linha na saída
2 ." Hello, Forth!" CR
3 BYE
```

Palavras

- Forth organiza seus códigos por meio de comandos nomeados, que abstram tarefas ou ações correlacionadas por meio de um nome comum
- Estes comandos nomeados seriam os equivalentes a funções em outras linguagem
- A sintaxe para a criação de um novo comando é

```
: nome implementação ;
```

  ou seja, inicia com dois-pontos, segue com o nome e continua com a definição, que termina com ponto-e-vírgula
- Comandos definidos dessa forma são denominados **palavras** (*words*)
- O padrão ANS da linguagem disponibiliza um conjunto grande (mais de 300) palavras pré-definidas, que podem ser usadas para definir novas palavras
- A habilidade de definir palavras por meio de palavras já definidas é denominada **extensibilidade**

Modo interativo

- No modo interativo do Forth (REPL), o interpretador responde aos comandos executados de forma bem sucedida com a palavra "ok"
- Por exemplo, entre no modo interativo e digite a tecla ENTER: Forth responderá "ok", movendo o cursor para a próxima linha
- Para inserir um ou mais números na pilha, basta inseri-los, na ordem desejada e separados por um espaço em branco, e digitar ENTER
- Para visualizar o estado atual da pilha, use a palavra **.s**, a qual não tem parâmetros

```
1 2 3 5 7 11
.s CR       \ <5> 2 3 5 7 11, <n> indica o tamanho da pilha
BYE
```

Palavras para a manipulação do terminal

- Quando uma palavra recebe um ou mais argumentos, eles são extraídos da pilha, do último para o primeiro
- Por exemplo, a palavra **spaces** recebe um argumento n e imprime n espaços no terminal

```
1 5 SPACES          \ Insere 5 na pilha e, em seguida, imprime 5 espaços no terminal
2 BYE
```

- A palavra **emit** recebe um inteiro n imprime no terminal o caractere cujo código ASCII é n:

```
1 42 EMIT CR        \ Imprime um asterisco no terminal
2 BYE
```

Palavras para a manipulação do terminal

- O código abaixo define uma nova palavra, chamada STAR, que imprime um asterisco no terminal:

```
1 : STAR   42 EMIT ;   \ A convenção é separar o nome da implementação com 3 espaços
2
3 STAR STAR STAR CR    \ Imprime uma nova linha composta de 3 asteriscos
4 BYE
```

- Também é possível definir uma nova palavra, chamada STARS, que recebe um argumento n e imprime n asteriscos consecutivos:

```
1 : STAR   42 EMIT ;
2 : STARS   0 DO  STAR  LOOP ;     \ As palavras DO e LOOP serão explicadas adiante
3
4 5 STARS CR                       \ Imprime uma nova linha com 5 asteriscos
5 BYE
```

O dicionário

- Todas as palavras definidas em Forth, seja em biblioteca padrão, seja pelo usuário, são armazenadas no "dicionário"
- Quando uma nova palavra é definida, Forth compila a palavra e a insere em seu dicionário
- Por exemplo, a linha abaixo é uma definição alternativa para a palavra `STAR`: (`[CHAR]` traduz o caractere para seu código ASCII):

```
: STAR   [CHAR] * EMIT ;
```

- Quando um comando é inserido no terminal, será ativada a palavra `INTERPRET`, que fará a leitura da entrada em busca de uma string (sequência de caracteres separada por espaços em branco)

O dicionário

- Se a palavra encontrada consta no dicionário, será ativada a palavra `EXECUTE`, que executa a definição da palavra e finaliza com a mensagem "`ok`"
- Se a palavra não consta no dicionário, então Forth tentará interpretar a string como um número: caso ele tenha sucesso na conversão, o número lido será inserido na pilha
- Se a palavra lida não é um número, Forth sinaliza um erro, indicando que a palavra não foi definida
- Os nomes das novas palavras devem ser compostos por, no máximo, 31 caracteres imprimíveis
- Por exemplo, a palavra `."`, que imprime no terminal a string que se segue, delimitada por aspas, é composta por dois símbolos de pontuação

Aritmética e a Pilha

- Conforme já mencionado, quando o interpretador Forth encontra um número, ele o armazena na pilha
- A pilha é uma estrutura de dados cuja política de acesso é a LIFO: *last in, first out*
- A cada instante, apenas o elemento do "topo" da pilha estará acessível
- Os operadores aritméticos (`+, -, *, /`) são palavras
- Ao serem executadas, estas palavras removem dois elementos do topo da pilha: na forma infixada, o primeiro elemento extraído será o operando à direita e o segundo elemento o operando à esquerda
- O resultado da operação é inserido na pilha

Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação:

Pilha:

Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação:

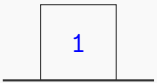
Pilha:

Exemplo de aritmética em Forth

Expressão: 1 2 3 + 4 * -

Operação:

Pilha:

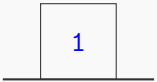


Exemplo de aritmética em Forth

Expressão: 1 2 3 + 4 * -

Operação:

Pilha:

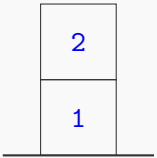


Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação:

Pilha:

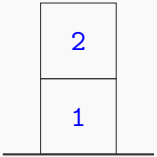


Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação:

Pilha:

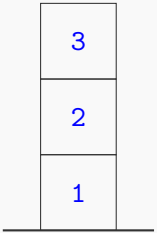


Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação:

Pilha:

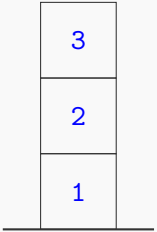


Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação:

Pilha:

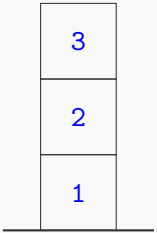


Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação: `+`

Pilha:

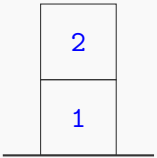


Exemplo de aritmética em Forth

Expressão: 1 2 3 **+** 4 ***** **-**

Operação: **+** 3

Pilha:

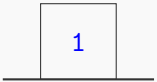


Exemplo de aritmética em Forth

Expressão: 1 2 3 **+** 4 ***** **-**

Operação: 2 **+** 3

Pilha:

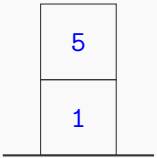


Exemplo de aritmética em Forth

Expressão: 1 2 3 + 4 * -

Operação: 2 + 3

Pilha:

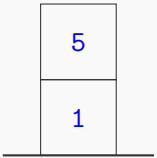


Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação:

Pilha:

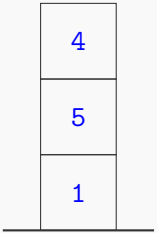


Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação:

Pilha:

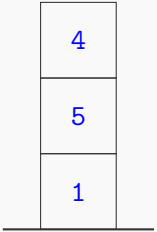


Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação:

Pilha:

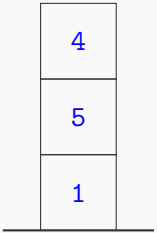


Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação: `*`

Pilha:

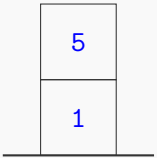


Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação: `* 4`

Pilha:

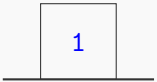


Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação: `5 * 4`

Pilha:

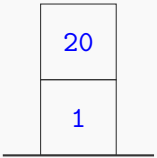


Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação: `5 * 4`

Pilha:

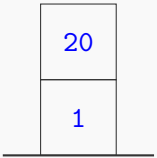


Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação:

Pilha:

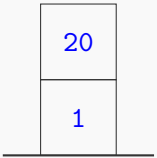


Exemplo de aritmética em Forth

Expressão: 1 2 3 **+** 4 ***** **-**

Operação: **-**

Pilha:

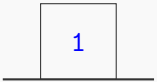


Exemplo de aritmética em Forth

Expressão: 1 2 3 + 4 * -

Operação: - 20

Pilha:



Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação: `1 - 20`

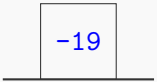
Pilha:

Exemplo de aritmética em Forth

Expressão: `1 2 3 + 4 * -`

Operação: `1 - 20`

Pilha:



Notação posfixa

- Forth utiliza de notação posfixa, isto, as palavras sucedem seus operandos, conforme ilustrado no exemplo anterior
- Esta convenção permite que a pilha seja preparada antes da execução de uma palavra e que as palavras extraiam seus argumentos, quando existirem, da pilha, em ordem reversa: do último para o primeiro argumento
- O código abaixo corresponde ao exemplo anterior: a palavra `.` (ponto final) extrai o topo da pilha e o imprime no terminal

```
1 1 2 3 + 4 * - . CR
2 BYE
```

- Observe que, ao final da execução, a pilha estará vazia

Stack underflow, stack overflow e stack effects

- A tentativa de se extrair um elemento quando a pilha está vazia resulta no erro *stack underflow*:

```
1 1 +      \ Stack underflow: não há argumentos suficientes para a adição
2 BYE
```

- Se não houver memória disponível para a inserção de um novo elemento na pilha, o interpretador emitirá o erro *stack overflow*
- Tanto as palavras já definidas por Forth quanto as palavras definidas pelo usuário podem tanto extrair elementos da pilha quanto inserir novos elementos
- Estas inserções ou remoções são denominadas *stack effects*
- Estes efeitos podem ser registrados por meio de um comentário que é inserido, na definição de uma nova palavra, entre o nome e a implementação

Notação de pilha

- O comentário que registra os *stack effects* da palavra é denominado **notação de pilha**
- A forma básica da notação de pilha é

```
( before -- after )
```

  onde *before* registra o que deve estar na *stack* antes da execução e *after* regista o que estará na *stack* após a execução
- A notação de pilha para a palavra **.** é

```
.   ( n --  )
```

- A notação de pilha para a palavra **+** é

```
+   ( n1 n2 -- sum )
```

Sumário

Palavra	Notação de pilha	Significado
`: name impl ;`	`( -- )`	Define a palavra `name` por meio das palavras `impl`
`CR`	`( -- )`	Imprime uma nova linha
`SPACES`	`( n -- )`	Imprime n espaços
`EMIT`	`( c -- )`	Imprime o caractere c
`."`	`( -- )`	Imprime a string delimitada por `"` que se segue
`.`	`( n -- )`	Imprime o número n, seguido de um espaço c
`+`	`( n1 n2 -- s )`	Computa $s = n1 + n2$
`-`	`( n1 n2 -- s )`	Computa $s = n1 - n2$
`*`	`( n1 n2 -- m )`	Computa $m = n1 \times n2$
`/`	`( n1 n2 -- q )`	Computa o quociente q da divisão inteira de $n1$ por $n2$
`MOD`	`( n1 n2 -- r )`	Computa o resto r da divisão inteira de $n1$ por $n2$
`/MOD`	`( n1 n2 -- q r )`	Computa o quociente q e o resto r da divisão de $n1$ por $n2$

Exemplo: conversão de tempo

```
1  \ Conversões para segundos
2  : HOURS    ( n -- n ) 3600 * ;
3  : MINUTES  ( n -- n ) 60 * ;
4  : SECONDS  ( n -- n ) ;
5
6  \ Versões no singular
7  : HOUR     HOURS ;
8  : MINUTE   MINUTES ;
9  : SECOND   SECONDS ;
10
11 \ Imprime o tempo correspondente a n segundos
12 : TIME ( n -- )
13    3600 /MOD . ." hour(s), "
14    60 /MOD . ." minute(s), "
15    . ." second(s)" ;
16
17 1 HOUR 30 MINUTES + 5000 SECONDS +
18 TIME CR          \ 2 hour(s), 53 minute(s), 20 second(s)
19 BYE
```

Exemplo: movimento retilíneo uniforme

```
\ Computa a posição final se o corpo, que estava inicialmente no km s0,
\ se desloca a v km/h durante t horas
: S ( s0 v t -- s )   * + ;

20 80 1 S . CR          \ 100
-14 125 3 S . CR        \ 361
BYE
```

Palavras para a manipulação da pilha

- Além da palavra **.**, que extrai o topo da pilha, Forth disponibiliza outras palavras para a manipulação da pilha
- A palavra **SWAP** *(a b -- b a)* inverte a ordem do topo com o segundo elemento da pilha:

```
1 1 2 3 . . . CR          \ 3 2 1
2 1 2 3 SWAP . . CR       \ 2 3 1
```

- A palavra **DUP** *(a -- a a)* duplica o elemento do topo da pilha:

```
1 1 2 3 dup               \ 1 2 3 3
```

Palavras para a manipulação da pilha

- A palavra **OVER** *(a b -- a b a)* insere uma copia o segundo elemento na pilha:

```
1 1 2 3 OVER              \ 1 2 3 2
```

- A palavra **ROT** *(a b c -- b c a)* rotaciona o terceiro elemento, de modo que ele passa a ocupar o topo da pilha:

```
1 1 2 3 4 ROT             \ 1 3 4 2
```

- A palavra **DROP** *(a --)* remove o topo da pilha, sem imprimí-lo na saída padrão

```
1 1 2 3 DROP              \ 1 2
```

Sumário

Palavra	Notação de pilha	Significado
`SWAP`	`( a b -- b a )`	Troca os dois elementos do topo da pilha de posição
`DUP`	`( a -- a a )`	Duplica o elemento do topo da pilha
`SPACES`	`( n -- )`	Imprime n espaços
`EMIT`	`( c -- )`	Imprime o caractere c
`."`	`( -- )`	Imprime a string delimitada por " que se segue
`.`	`( n -- )`	Imprime o número n, seguido de um espaço c
`+`	`( n1 n2 -- s )`	Computa $s = n1 + n2$
`-`	`( n1 n2 -- s )`	Computa $s = n1 - n2$
`*`	`( n1 n2 -- m )`	Computa $m = n1 \times n2$
`/`	`( n1 n2 -- q )`	Computa o quociente q da divisão inteira de $n1$ por $n2$
`MOD`	`( n1 n2 -- r )`	Computa o resto r da divisão inteira de $n1$ por $n2$
`/MOD`	`( n1 n2 -- q r )`	Computa o quociente q e o resto r da divisão de $n1$ por $n2$

Referências

1. **BRODIE**, Leo. Starting FORTH, Online Edition, acesso em 10/10/2025.
2. **HORSE**, M. D. Learn X in Y minutes - Where X is Forth, acesso em 10/10/2025.
3. **MORGAN**, Nick. Easy Forth, acesso em 10/10/2025.
4. **NASA**. Forth in Space Applications, acesso em 10/10/2025.
5. **Wikipédia**. Stack-oriented programming, acesso em 13/10/2025.