Programação na HP Avançadíssima

Daniel Rossato

Henrique Camargo

Conteúdo

- Manipulação de pilha
- Condicionais

Manipulação de Pilha

- DUP
  - Duplica elemento
- DUP2
  - Duplica 2 elementos
- DUPN
  - Duplica *n* elementos
- DROP
  - Remove elemento
- DROP2
  - Remove 2 elementos
- DROPN
  - Remove *n* elementos

Manipulação de Pilha

- OVER
  - Copia segundo elemento da pilha
- PICK
  - Copia o *n*-ésimo elemento da pilha
- SWAP
  - Troca a ordem dos 2 primeiros elementos
- ROLL
  - Gira a pilha pra cima a partir do elemento *n*
- ROLLD
  - Gira a pilha pra baixo a partir do elemento *n*
- DEPTH
  - Retorna o número de elementos da pilha

Manipulação de Pilha

- Exemplos
  - << 3 DUPN + + * * SWAP - >>
  - << DUP2 * 3 ROLLD + / >>
  - << DEPTH PICK >>
  - << 3 DUPN DROP2 >>
  - << DEPTH PICK OVER DEPTH ROLLD DEPTH ROLLD DEPTH 2 - DROPN >>

Manipulação de Pilha

- Exemplo
  - << DUP2 DUP2 SWAP - 3 ROLLD SQ SWAP SQ + SWAP - DUP 3 ROLLD * 3 ROLLD * SWAP >>

  - Resultado:

$$x \cdot \left(y^2 + x^2 - (y - x)\right)$$

$$y \cdot \left(y^2 + x^2 - (y - x)\right)$$

Condicionais

- Avalia uma expressão, e executa uma ação conforme o resultado
  - 0 == false
  - 1 == true (ou outros valores)
- Operadores lógicos
  - == (Não é “=“ !!)
  - /= (símbolo “diferente”)
  - AND, OR, XOR, NOT
  - <, >, <=, >= (símbolos de “maior-igual”, etc)
- Também são inseridos no modo RPN

Condicionais

- Exemplo
  - 3 4 > => false
  - 1 5 < => true
  - 2 0 == => false
  - 1 0 OR => true
  - 2 2 /= 3 0 > AND => false
  - 1 0 AND 1 0 OR OR => true

Condicionais

- Bloco IF
  - Estruturalmente não recebe seus parâmetros em RPN
  - IF *expressão* THEN *comandos* END
    - << -> A B
      << **IF** A B <
      THEN "A é menor que B!"
      END
      >>
      >>

Condicionais

- Bloco IF-ELSE
  - IF *condicao* THEN *comando1* ELSE *comando2* END
  - << -> A B

```
<< IF A B <
        THEN “A é menor que B!”
        ELSE “A não é menor que B!”
        END
 >>
>>
```

Condicionais

- Blocos IF podem ser encadeados
- Exemplo: devolve o menor de 3 números
- << -> A B C

```
        << IF A B < A C < AND
                THEN A “A é menor” ->TAG
                ELSE
                        IF B C <
                                THEN B “B é menor” ->TAG
                                ELSE C “C é menor” ->TAG
                        END
                END
        >>
>>
```

Condicionais

- Bloco CASE
  - CASE
    condicao1 THEN *comando1* END
    condicao2 THEN *comando2* END
    condicao3 THEN *comando3* END
    comandosDefault
    END

Condicionais

```
<< -> x
    <<    CASE
              x 1 == THEN “Um” END
              x 2 == THEN “Dois” END
              x 3 == THEN “Tres” END
              x 4 == THEN “Quatro” END
              x 5 == THEN “Cinco” END
              “Sei la”
          END
          x SWAP ->TAG
    >>
>>
```

Laços

- Blocos de código repetidos conforme uma condição, ou um determinado número de vezes
- Ex.: While, Do, For, etc.
- Como as condicionais, mesma estrutura de linguagem C e afins

Laços

- Bloco WHILE
  - WHILE *condição* REPEAT *comandos* END
  - Repete os comandos enquanto a condição for verdadeira
- Bloco UNTIL
  - DO *comandos* UNTIL *condição* END
  - Repete enquanto a condição for falsa
  - Sempre vai executar uma vez, mesmo que a condição seja verdadeira no início.

Laços

- Exemplo
  - << -> A B C
    << A ‘X’ STO **WHILE** X C < **REPEAT**
    X B + ‘X’ STO
    END
    >>
    >>

Laços

- Exemplo
  - << -> A B
    <<
    >>
  >>

Laços

- Laço START
  - *Inicial final* START *comandos* NEXT
  - Repete os comandos *final-inicial* vezes
  - Não fornece acesso à variável de iteração
- Exemplo
  - << 1 SWAP START “Oi!” 2200 0.3 BEEP NEXT >>

Laços

- Laço FOR
- *start stop* FOR *variavel comandos* NEXT
- Em vez de NEXT, pode-se usar STEP
- Em vez de aumentar a variável em 1, aumenta com o valor da pilha

Laços

- Exemplo
  - << 1 SWAP FOR k 3 k * “opa” ->TAG NEXT >>
  - << 1 SWAP FOR k 3 k * “opa” ->TAG 2 STEP >>

Laço

- Exercício
  - Escreva um programa que calcule o valor de pi com precisao de 0.001

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \ldots$$

Matrizes

- Comandos de manipulação
- GET
  - Retorna o *n*-ésimo elemento da lista
- PUT
  - Coloca na posição *n* o elemento *x*
- SIZE
  - Retorna uma lista no formato {linhas colunas}
- TRAN
  - Transpõe a matriz
- Há comandos para montar matrizes, mas é mais fácil montar uma lista e converter com o comando AXL

Matrizes

- Exercício
  - Faça um programa que receba uma matriz e duas coordenadas, e retorne o elemento nesta posição
  - Faça um programa que receba uma uma matriz, duas coordenadas, um valor, e coloque este valor na matriz na posição das coordenadas

Listas

- Definidas por chaves { }
- Montada pelo comando ->LIST
  - Recebe o número de elementos da lista
  - Exemplo:
  - << 10 20 30 40 50 5 ->LIST >>
  - Resultado: {10 20 30 40 50}
- EVAL desmancha a lista

Listas

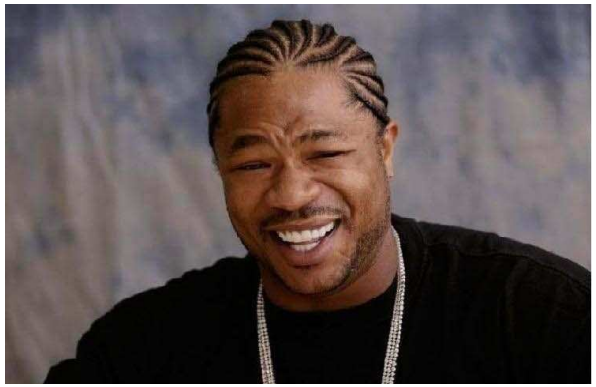
- ADD
  - Adiciona um elemento à lista
- SORT
  - Ordena a lista
- AXL
  - Converte uma matriz em uma lista de listas, e vice-versa { {1 2 3} {4 5 6} {7 8 9} }
- ->LIST
  - Cria uma lista com os *n* elementos na pilha

Listas

- Exercício
  - Faça um programa que crie uma lista de 9 elementos, de 1 a 5
  - Faça um programa que receba um número, e monte uma matriz identidade com lado igual a esse número

Gráficos

- Gráfico em barra
  - Comando BARPLOT
    - Plota os pontos na matriz-coluna “ΣDAT”
  - Para fazer uma matriz coluna:
    - Crie uma lista
    - Coloque a lista dentro de outra lista
    - Converta para matriz com AXL
    - Transponha com TRAN



Gráficos

- Exercício
    - Monte uma lista com 2 ciclos de uma senóide de amplitude 3, e ruído de amplitude 1, tendo 65 amostras