Núcleo de Informática Aplicada à Educação
Universidade Estadual de Campinas

RESUMO

Como o próprio título sugere este texto apresenta um panorama geral da linguagem de programação Logo. Neste, são mostradas de forma conjunta várias primitivas relacionadas à programação gráfica e simbólica, com o intuito de minimizar a segmentação existente entre o "universo da Tartaruga" e o "universo das listas". Embora o material possua um caráter amplo, a abordagem de cada primitiva procura caracterizar suas peculiaridades, sua funcionalidade, incorporando um linguajar mais técnico que auxilia o entendimento de alguns aspectos da atividade de programação. O material possui o conteúdo acima descrito em forma de texto e, também algumas sugestões de projetos sem, no entanto, esgotá-los. Estes servem para desencadear, no leitor, idéias de novos projetos. Embora os exemplos do material utilizem a implementação do Superlogo versão 3.0, todos eles podem ser definidos usando-se outras versões do Logo.

NIED - Memo Nº 35
2000

Tartaruga, Figuras, Palavras, Listas e Procedimento:
Um primeiro passeio pelo Logo – SuperLogo 3.0

Heloísa Vieira da Rocha (e-mail: heloisa@dcc.unicamp.br)
Fernanda M. P. Freire (e-mail: ffreire@unicamp.br)
Maria Elisabette B.B. Prado (e-mail: bprado@unicamp.br)

Núcleo de Informática Aplicada à Educação - Nied
Universidade Estadual de Campinas - Unicamp
Cidade Universitária "Prof. Zeferino Vaz"
Prédio V da Reitoria – 2º Piso
13083-970 - Campinas - SP
Telefones: ++ 55 (19) 788-8136 ou 788-7350
Fac-símile: ++ 55 (19) 788-8136 ou 788-7350 (Ramal 30)

CONTEÚDO

Tartaruga, Figuras, Palavras, Listas e Procedimento: Um primeiro passeio pelo Logo

APRESENTAÇÃO

Este texto é resultado de nossa experiência no fornecimento de diversos cursos de programação Logo aos mais distintos usuários.

É fato conhecido que o aprendizado de programação Logo é visto como necessariamente dividido em duas etapas: Logo gráfico e Logo simbólico. Isto tem conduzido à uma estrutura de cursos bastante interessante. Existe um curso de Logo onde somente se trabalha no universo gráfico da linguagem, e quando muito se introduz objetos de animação. Em seguida, quando solicitado, é apresentado um curso denominado Logo Avançado, onde são trabalhadas as operações com palavras e listas. Isto tem gerado um senso comum de que a linguagem Logo é dividida em partes bastante diferentes e raramente, pode-se ver projetos englobando os assim divididos universos da linguagem, que na realidade são um só. E com isso, uma programação mais avançada de Logo raramente é possível.

É também fato conhecido a extrema dificuldade de se transpor a barreira criada entre as duas partes da linguagem. A maioria das pessoas que programam Logo dificilmente programam problemas que não sejam gráficos. Assim, obtêm-se uma visão distorcida do potencial de Logo enquanto uma linguagem de programação de propósito geral.

Por esta razão, iniciamos em nossos cursos, com bastante sucesso, uma abordagem integrada da linguagem. O curso inicial de Logo não se restringe mais a comandos gráficos e, sim, oferece um "passeio" significativo por todos os comandos e operações da linguagem. Neste "passeio" procuramos ressaltar as diferenças conceituais entre as primitivas sem, contudo, separá-las em dois grandes conjuntos.

Outro aspecto que também acrescentamos aos nossos cursos foi um tratamento mais técnico da linguagem. Por ser Logo uma linguagem essencialmente dirigida a aprendizes, é comum verificar que ela, mesmo em cursos de formação de profissionais, é tratada com muita informalidade e, muitas vezes, com uma certa infantilidade. Acreditamos que não se pode ignorar que Logo é computacional, e portanto formal e técnica. Ignorar esses aspectos dificulta

o entendimento de muitos conceitos inerentes a qualquer linguagem de programação. Por exemplo, é muito comum programadores de Logo terem como conceito de variável somente o de parâmetro. Comprovamos que a introdução precoce de conceitos como a natureza da primitiva, natureza dos parâmetros, ordem de execução de operações, etc. não dificultam em nada o iniciante em programação, pelo contrário, facilitam o entendimento do funcionamento da linguagem em que estão aprendendo a se expressar.

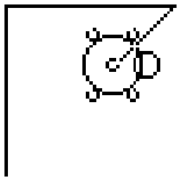
Este texto é a primeira parte do material de apoio que estamos utilizando para atender estes objetivos. É importante frisar que simplesmente o uso do texto não irá viabilizar o que pretendemos. Existe associada toda uma postura pedagógica, pois de forma alguma pretendemos menosprezar a dificuldade que é programar com autonomia em uma linguagem que possui, pelo menos, dois diferentes paradigmas.

Manipulando a Tartaruga

A tartaruga é um cursor gráfico que aparece no centro da tela gráfica. Para fazer desenhos basta movimentá-la na tela de modo que ela deixe traços pelo seu caminho. Há quatro comandos básicos que movimentam a tartaruga. Os comandos PARAFRENTE nº (PF nº) e PARATRÁS nº (PT nº) fazem a tartaruga andar e os comandos PARADIREITA nº (PD nº) e PARAESQUERDA nº (PE nº) giram a tartaruga[1]. Ao usar esses comandos é necessário especificar o número de passos ou a medida do grau do giro.

[1] A maior parte dos comandos do Logo possui uma forma abreviada que simplifica a digitação. Após a apresentação de cada comando adotaremos a forma abreviada do mesmo. Para maiores detalhes consulte o manual de referência da implementação usada.

Observe a seqüência dos comandos e acompanhe o efeito da mesma[2].



```
TAT
PF 50 PD 90
PF 50 PE 45
PT 25 PD 45
```

A tartaruga é definida por uma **posição** em relação a um sistema de coordenadas cartesianas (x, y) cujo ponto [0 0] representa o centro da tela gráfica e por uma **orientação** em relação a um eixo imaginário cujo ponto inicial é 0°. Os comandos PF e PT alteram a posição da tartaruga e os comandos PD e PE a sua orientação.

Os números que seguem os comandos PF, PT, PD, PE são chamados de parâmetros ou entradas. Existem comandos em Logo que não precisam de parâmetro como o comando TARTARUGA (TAT). Da mesma forma, há comandos que precisam de mais de um parâmetro. No exemplo mostrado os parâmetros usados são números mas, um parâmetro pode ser também uma palavra ou uma lista como veremos adiante. A omissão de um parâmetro quando ele é necessário produz uma mensagem de erro.

Para movimentar a tartaruga sem deixar traços usa-se o comando USENADA (UN) seguido de um comando que desloca a tartaruga. Analogamente, para apagar um traço na tela existe o comando USEBORRACHA (UB). Para retornar ao traço digita-se o comando USELÁPIS (UL). O comando DESAPAREÇATAT (DT) torna a tartaruga invisível e o comando APAREÇATAT (AT) faz retornar a sua figura. Ambos são úteis durante a realização de um desenho. Se for necessário recomeçar um desenho ou iniciar um novo pode-se usar o comando TAT que limpa a tela e recoloca a tartaruga na sua posição e orientação originais.

[2] Os exemplos que serão apresentados usam como referência o software SuperLogo versão 3.0. As primitivas acentuadas deverão ser digitadas com letras minúsculas.

As figuras abaixo mostram o uso desses comandos.

```
TAT
PF 10
UN
PF 20
```

```
DT
UL
PF20
```

```
UB
PT 20
AT
```

```
TAT
```

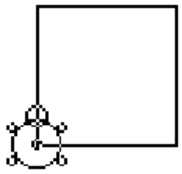
A cor do traço da tartaruga pode ser mudada alterando-se a cor do lápis da tartaruga através do comando MUDECL nº que significa mude a cor do lápis. Para se obter um melhor contraste pode-se modificar a cor da tela por meio do comando MUDECF nº que significa mude a cor do fundo. Cada número usado como parâmetro desses comandos corresponde a uma cor pré-estabelecida pelas diferentes implementações da linguagem Logo.

Repetindo uma seqüência de ações

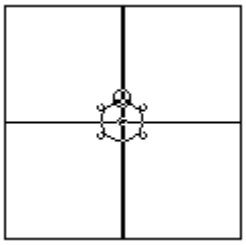
Um comando bastante útil do repertório do Logo é o comando REPITA. Ele é usado quando se quer efetuar uma mesma ação, ou seqüência de ações, um determinado número de vezes. O REPITA precisa de dois parâmetros: um número e uma lista. O número refere-se ao número de vezes que a lista deve ser repetida e a lista refere-se a ação que deve ser realizada. Portanto, a forma genérica desse comando é:

REPITA <*número*> <*lista*>

O comando REPITA pode ser usado em vários contextos, entre eles, para desenhar figuras geométricas.



REPITA 4 [PF 50 PD 90]



REPITA 4 [REPITA 4 [PF 50 PD 90] PD 90]

Outro exemplo de uso do comando REPITA é a execução do comando

REPITA 4 [MUDECF 5 ESPERE 200 MUDECF 0 ESPERE 200]

que provoca mudanças alternadas da cor do fundo da tela.

Outras Possibilidades do Logo

Como vimos o lápis da tartaruga pode assumir diferentes cores. Com este recurso pode-se pintar áreas perfeitamente delimitadas da tela, através do comando PINTE. Para pintar é necessário seguir alguns passos. Veja um exemplo de como proceder para pintar o interior de um quadrado:

```
REPITA 4 [ PF 60 PD 90 ]
UN PD 45 PF 20
UL MUDECP 0 PINTE
```

Observe que para pintar a tartaruga deve ser levada, sem lápis, para dentro da região a ser pintada. Em seguida para pintar ela deve estar usando lápis com a cor de preenchimento desejada. Para mudar somente a cor da linha de uma figura, a tartaruga deve estar sobre a mesma.

Escrevendo na Janela Gráfica

Pode-se escrever na tela através do comando ROTULE. Este comando precisa de um parâmetro que pode ser um número, ou uma palavra ou uma lista. Por exemplo, os comandos:

```
ROTULE "LOGO
UN PT 20
ROTULE [ Maria gosta de José ]
PT 20
ROTULE 1998
```

produziriam como resultado:

LOGO
Maria gosta de José
1998

Observe que quando a entrada do comando ROTULE é uma lista os colchetes não aparecem impressos na Janela Gráfica. O comando ROTULE só imprime o conteúdo da lista usada como parâmetro.

Compreendendo a natureza dos parâmetros

Como já foi dito, existem comandos que precisam de parâmetro. Esses parâmetros possuem diferentes naturezas: podem ser números, palavras ou listas.

Palavra é uma seqüência de caracteres precedida por aspas ("). Como por exemplo:

"casa "123 "dx40 "345F

Número é um tipo especial de palavra, que dispensa o uso de aspas para facilitar as operações aritméticas. Um número é constituído somente por dígitos, podendo ser um valor inteiro ou real (com ponto decimal). Exemplo:

123 354.5 1.5 48

Lista é um conjunto de palavras, números ou listas escrito entre colchetes ([ ]). Como por exemplo:

[abacaxi casa 32]
Esta lista possui 3 elementos: os dois primeiros são palavras e o último é número; ou, todos são palavras. Observe que quando uma palavra é elemento de uma lista não precisa vir precedida por aspas.

[[pastel] hamburger pizza]
O elemento de uma lista também pode ser uma lista como é o caso do primeiro elemento desse exemplo.

[[coca cola] guaraná suco]
Esta lista possui 3 elementos: o primeiro é uma lista e os dois últimos são palavras. Neste caso o primeiro elemento da lista é também uma lista de dois elementos que são palavras.

Existe a definição de lista vazia que é denotada por [ ] e palavra vazia que é denotada por aspas seguida por um espaço em branco: " .

Conhecendo as Operações

Operações são primitivas que possibilitam a passagem de informações. As operações retornam valores que são utilizados por comandos ou por outras operações. Há operações que lidam com o universo da Tartaruga, outras que manipulam exclusivamente números e que são denominadas de operações aritméticas, e outras ainda, que lidam com palavras e listas. Vejamos alguns exemplos de cada uma delas:

MOSTRE CF
O comando MOSTRE (MO) imprime na Janela de Comandos o resultado da operação CORDOFUNDO (CF) que é um número que corresponde à cor do fundo da Janela Gráfica.

MOSTRE CL
O comando MOSTRE imprime na Janela de Comandos o resultado da operação CORDOLÁPIS (CL) que é uma lista com número correspondentes à última cor assumida pelo lápis da tartaruga.

MOSTRE DÇ
O comando MOSTRE imprime na Janela de Comandos o resultado da operação DIREÇÃO (DÇ) que é um número que corresponde à última orientação da tartaruga.

MOSTRE POS
O comando MOSTRE imprime na Janela de Comandos o resultado da operação POSIÇÃO (POS) que é uma lista que corresponde à posição adotada pela tartaruga de acordo com o sistema de coordenadas cartesianas (eixo x e eixo y respectivamente).

As operações aritméticas como: + - * / precisam de parâmetros sempre numéricos e produzem como resultado, também, um número. Por exemplo:

PF 50 * 2

O comando PF desloca a tartaruga 100 passos que corresponde ao resultado da operação de multiplicação de 50 por 2

REPITA 6 [PT 50 PE 360/6]

O comando PE gira a tartaruga 60 graus que corresponde ao resultado da operação de divisão.

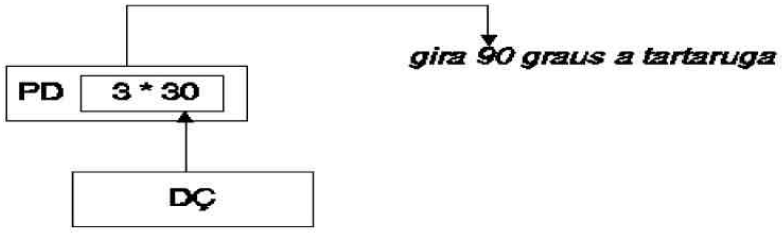
PT 5 + 6 + 15 + 5

O comando PT desloca a tartaruga 31 passos que é o resultado das três operações de adição.

PD 3 * DÇ

O comando PD gira a tartaruga um determinado número. Este número é o resultado da operação de multiplicação que, por sua vez, usa como um dos parâmetros o resultado da operação DÇ. [3]

Observe como o interpretador Logo faz a avaliação dessa instrução, supondo-se que a tartaruga estivesse orientada em 30°.



Há um extenso conjunto de operações que lidam com palavras e listas. Ele pode ser subdividido de acordo com diferentes funções: operações que separam elementos, que buscam e contam elementos, que constróem estruturas e que concatenam elementos. A seguir apresentaremos as diferentes operações de acordo com as suas funções:

[3] Geralmente em Logo, usa-se parênteses para modificar a ordem de prioridade dos operadores.

Separando Elementos

Existe, basicamente, dois tipos de operações que facilitam a separação de elementos de palavras e listas. Operações que retornam o primeiro ou o último elemento de uma palavra ou de uma lista: PRIMEIRO (PRI) e ÚLTIMO (ULT). E, operações que retornam o restante de uma palavra ou de uma lista sem o primeiro ou sem o último elemento da palavra ou da lista: SEMPRIMEIRO (SP) e SEMÚLTIMO (SU). Essas operações não aceitam palavras ou listas vazias como parâmetro. Veja alguns exemplos:

MOSTRE PRI "abacaxi
O comando MOSTRE imprime na Janela de Comandos o resultado da operação PRI que é a palavra a

MOSTRE PRI [a b c]
O comando MOSTRE imprime na Janela de Comandos o resultado da operação PRI que é a palavra a

MOSTRE SP "abacaxi
O comando MOSTRE imprime na Janela de Comandos o resultado da operação SP que é a palavra bacaxi

MOSTRE SP [a b c]
O comando MOSTRE imprime na Janela de Comandos o resultado da operação SP que é a lista [b c]

MOSTRE SU “amora
O comando MOSTRE imprime na tela o resultado da operação SU que é a palavra amor

MOSTRE PRI 5476
O comando MOSTRE PRI 5476 imprime na Janela de Comandos o resultado da operação PRI que é o número 5

MOSTRE SU 6791

O comando MOSTRE imprime na Janela de Comandos o resultado da operação SU que é o número 679

MOSTRE SU [2 4 6]

O comando MOSTRE imprime na Janela de Comandos o resultado da operação SU que é a lista [2 4]

Observe que as operações SP e SU retornam valores que são sempre da mesma natureza do seus parâmetros. Isto é, se o parâmetro da operação SP for uma palavra o resultado será uma palavra e se o parâmetro for uma lista, o resultado será uma lista.

Pode-se também, combinar essas operações para obter operações mais sofisticadas. Suponha que se queira imprimir a letra **m** da palavra **ameixa**. Pode-se combinar as operações da seguinte forma:

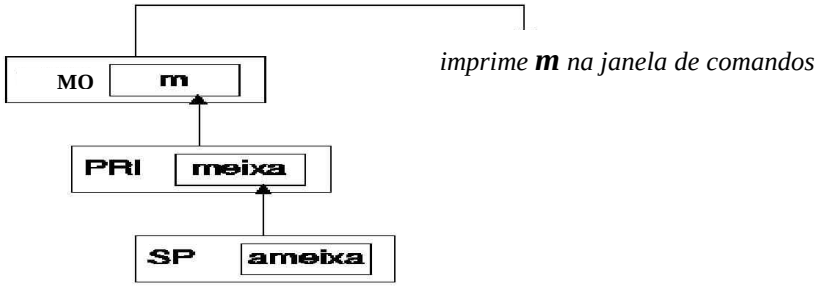
MOSTRE PRI SP "ameixa

A forma de execução deste comando pode ser descrita pela seguinte seqüência de passos:

1. O interpretador reconhece o comando MOSTRE e sabendo que ele precisa de um parâmetro continua lendo a instrução...
2. Em seguida o interpretador encontra a operação PRI e reconhece que o seu resultado será o parâmetro do comando MOSTRE.
3. O interpretador sabe que a operação PRI precisa de um parâmetro e continua então lendo a instrução sob o controle da operação PRI.
4. A próxima palavra encontrada é SP que é reconhecida como o nome de uma operação que também precisa de um parâmetro. O interpretador continua lendo sob o controle da operação SP. "Observe que MOSTRE e PRI estão suspensos aguardando os seus parâmetros."
5. É encontrada a palavra **ameixa** que é reconhecida como parâmetro da operação SP.
6. A operação SP é resolvida retornando a palavra **meixa**.
7. Este valor é passado para a operação PRI como seu parâmetro. A operação PRI é efetivada retornando o valor **m**.

8. Este valor é o parâmetro de MOSTRE que é então executado imprimindo na Janela de Comandos a palavra **m**.

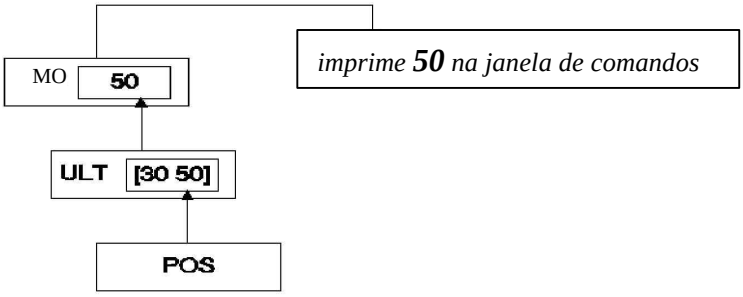
Veja a representação dessa avaliação:



Pode-se obter o valor da coordenada y[4] da tartaruga através da operação POS. Para tanto é necessária uma combinação de operações do tipo:

MOSTRE ULT POS

Esquematicamente e supondo-se que a tartaruga esteja na posição [30 50], a avaliação do comando seria feita pelo interpretador Logo da seguinte maneira:



Ou ainda, para se obter a letra **c** da lista **[a b [c d e] f]** pode-se fazer:

MOSTRE PRI PRI SP SP [a b [c d e] f]

ou

MOSTRE PRI SU SU ULT SU [a b [c d e] f]

Um aspecto importante dessa forma de notação é o mecanismo de avaliação que precisa ser bem compreendido para se operar com palavras e listas de modo geral. Observe que todas as operações em Logo, com exceção das aritméticas, são pré-fixas, ou seja, o operador é escrito antes dos respectivos operandos.

Buscando e Contando Elementos

Para selecionar um determinado elemento de uma palavra ou de uma lista usa-se a operação ELEMENTO. Essa operação precisa de dois parâmetros: um número e uma palavra ou lista. O número indica a posição do elemento desejado da palavra ou da lista.

ELEMENTO número *<palavra ou lista>*

O número indica a posição do elemento desejado da palavra ou da lista. Veja alguns exemplos:

MOSTRE ELEMENTO 2 [casa [pneu carro]]

O comando MOSTRE imprime na Janela de Comandos o resultado da operação ELEMENTO que é a lista [pneu carro]

MOSTRE ELEMENTO 5 "boneca

O comando MOSTRE imprime na Janela de Comandos o resultado da operação ELEMENTO que é a palavra c

Quando se deseja saber o número total de elementos de uma palavra ou lista pode-se usar a operação NUMELEM. Essa operação só tem um parâmetro que é o próprio objeto.

MOSTRE NUMELEM [[abc]]

O comando MOSTRE imprime na Janela de Comandos o resultado da operação NUMELEM que é o número 1

MOSTRE NUMELEM "abc

4 A operação COORY n° retorna o valor da coordenada y da tartaruga .

O comando MOSTRE imprime na Janela de Comandos o resultado da operação NUMELEMENTO que é o número 3.

Construindo Estruturas

Pode-se construir estruturas do tipo palavras através da operação PALAVRA (PAL) e do tipo listas através das operações LISTA e SENTENÇA (SN). Todas essas operações precisam de pelo menos dois parâmetros[5]. A seguir apresentamos exemplos de cada uma dessas operações:

MOSTRE PAL "cachorro "quente
O comando MOSTRE imprime na Janela de Comandos o resultado da operação PAL que é a palavra cachorroquente

MOSTRE (PAL 34 57 28)
O comando MOSTRE imprime na Janela de Comandos o resultado da operação PAL que é a palavra 345728
A operação PAL precisa de parâmetros que sejam palavras.

A diferença entre as operações LISTA e SN é bastante sutil. Por essa razão apresentaremos o mesmo exemplo para as duas operações para facilitar a comparação entre elas. Ambas as operações aceitam tanto listas quanto palavras como entrada.

MOSTRE LISTA 45 80
O comando MOSTRE imprime na Janela de Comandos o resultado da operação LISTA que é a lista [45 80]

MOSTRE SN 45 80
O comando MOSTRE imprime na Janela de Comandos o resultado da operação SN que é a lista [45 80]

5 Pode-se aumentar o número de entradas dessas operações usando-se parênteses antes de digitar o nome da operação e após a última entrada.

A operação LISTA coloca em uma lista os parâmetros usados mantendo a natureza dos mesmos. Nesse exemplo, a operação SN é idêntica. O seu resultado é uma lista cujos elementos são as palavras usadas como entradas. Veja outros exemplos:

MOSTRE (LISTA [tudo] [bem] [José])
O comando MOSTRE imprime na Janela de Comandos o resultado da operação LISTA que é a lista [[tudo] [bem] [José]]

MOSTRE (SN [tudo] [bem] [José])
O comando MOSTRE imprime na Janela de Comandos o resultado da operação SN que é a lista [tudo bem José]

MOSTRE LISTA [Bom] "Dia
O comando MOSTRE imprime na Janela de Comandos o resultado da operação LISTA que é a lista [[Bom] Dia]

MOSTRE SN [Bom] "Dia
O comando MOSTRE imprime na Janela de Comandos o resultado da operação SN que é a lista [Bom Dia]

Observe que nestes exemplos operação LISTA não altera a natureza das entradas; nesse exemplo, elas continuam sendo lista e palavra enquanto elementos do resultado final. O mesmo não acontece com a operação SN; o resultado final apresenta elementos que são palavras, independentemente da natureza original dos parâmetros. Mais exemplos:

MOSTRE LISTA [8] [3 [6]]
O comando MOSTRE imprime na Janela de Comandos o resultado da operação LISTA que é a lista [[8] [3 [6]]]

MOSTRE SN [8] [3 [6]]
O comando MOSTRE imprime na Janela de Comandos o resultado da operação SN que é a lista [8 3 [6]]

Note como a operação SN trata os parâmetros quando esses são listas constituídas por sub-listas. A operação retorna uma lista cujos elementos são palavras e listas. O parâmetro que originariamente era lista torna-se uma palavra da lista final. A sub-lista do parâmetro original torna-se uma lista na lista final.

Resumindo, a operação LISTA retorna uma lista de suas entradas e a operação SN retorna uma lista dos elementos de suas entradas.

Concatenando Elementos

Qualquer palavra ou lista pode ser adicionada a uma outra lista. A operação JUNTENOINÍCIO (JI) adiciona uma palavra ou lista no início de uma determinada lista e a operação JUNTENOFIM (JF) adiciona uma palavra ou lista no fim de uma dada lista. Portanto, o primeiro parâmetro dessas operações pode ser uma palavra ou lista e o segundo é sempre uma lista. O resultado final dessas operações é sempre uma lista. Vejamos alguns exemplos:

MOSTRE JF [coca] [[cachorro quente] hamburger pastel]
O comando MOSTRE imprime na Janela de Comandos o resultado da operação JF que é a lista [[cachorro quente] hamburger pastel [coca]]

MOSTRE JI "fim [[começo]]
O comando MOSTRE imprime na Janela de Comandos o resultado da operação JI que é a lista [fim [começo]]

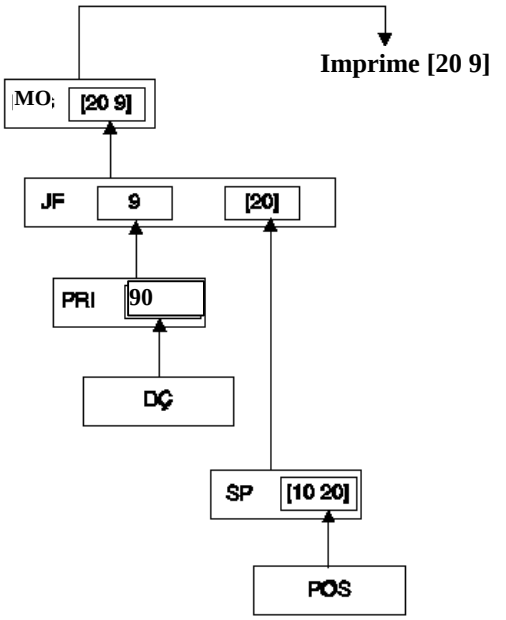
MOSTRE JF "início [ ]
O comando MOSTRE imprime na Janela de Comandos o resultado da operação JF que é a lista [início]

MOSTRE JI 30 [de março de 1961]
O comando MOSTRE imprime na Janela de Comandos o resultado da operação JI que é a lista [30 de março de 1961]

MOSTRE JF PRI DÇ SP POS

O comando MOSTRE imprime na Janela de Comandos o resultado da operação JF que é uma lista.

Veja a avaliação desta instrução supondo que a tartaruga está na posição [10 20] com a direção de 90°:



Conhecendo os Predicados

São denominados de predicados as operações que retornam valores booleanos identificados pelas palavras VERD e FALSO que representam verdadeiro e falso respectivamente. Por exemplo ÉVISÍVEL é uma operação que retorna a palavra VERD se a tartaruga está visível na tela e FALSO caso contrário e, portanto, é um predicado.

Dentre as operações de manipulação de palavras e listas existem predicados definidos para saber se um determinado objeto é uma palavra, uma lista, um elemento de uma estrutura etc.

Por exemplo, ÉLISTA é um predicado que tem um parâmetro que pode ser uma palavra ou uma lista e retorna VERD se for lista e FALSO caso contrário.

ÉPALAVRA é outro predicado que tem um parâmetro, que pode ser uma palavra ou lista. Retorna VERD se o parâmetro for uma palavra e FALSO se for uma lista. Com funcionamento análogo existe o predicado ÉNÚMERO que verifica se o parâmetro é um número ou não.

ÉVAZIA é também um predicado que tem como parâmetro uma palavra ou lista. Retorna VERD se o parâmetro for uma palavra vazia ou uma lista vazia e FALSO caso contrário.

MOSTRE (ÉLISTA [abacaxi])

Imprime na Janela de comandos o resultado da operação ÉLISTA [abacaxi], que é *Verd*

MOSTRE (ÉLISTA “abacaxi)

Imprime na Janela de comandos o resultado da operação ÉLISTA “abacaxi, que é *Falso*

Conhecendo os Operadores Lógicos e Relacionais

Os operadores lógicos: OU, E e NÃO operam sobre os predicados e produzem VERD ou FALSO de acordo com a seguinte tabela, denominada Tabela Verdade:

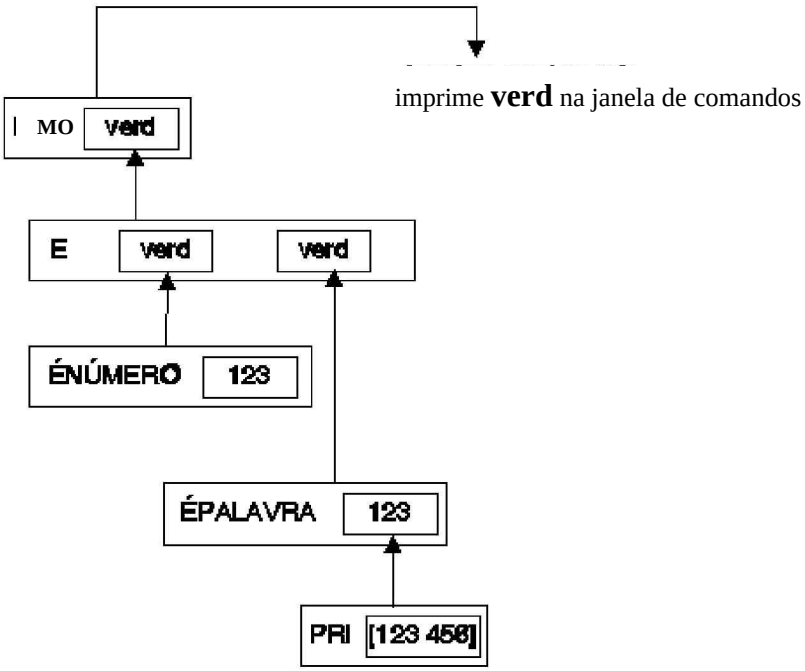
PRED 2	PRED 1	OU	E	NÃO PRED 1
F	F	F	F	V
F	V	V	F	V
V	F	V	F	F
V	V	V	V	F

A leitura desta tabela se faz da seguinte maneira: considerando a primeira linha, tem-se que se PRED 1 (predicado 1) é FALSO e PRED 2 (predicado 2) também é FALSO, a operação OU aplicada aos dois predicados retorna o valor FALSO, a operação E também retorna FALSO e a operação NÃO PRED 1 retorna VERD. Observe que as operações OU e E são binárias, ou seja, precisam de dois operandos, e a operação NÃO é unária, precisa de um operando.

MOSTRE E (ÉNÚMERO 123) (ÉPALAVRA PRI [123 456])

O comando MOSTRE imprime na Janela de Comandos a o resultado do operador lógico E que é a palavra VERD. Neste caso os parênteses servem para clarificar a operação desejada.

Acompanhe este esquema de avaliação:



MOSTRE OU (ÉLISTA "ana) (ÉNÚMERO [680])

O comando MOSTRE imprime na Janela de Comandos o resultado do operador lógico OU que é a palavra FALSO

MOSTRE NÃO ÉLISTA [5 6 7]

O comando MOSTRE imprime na Janela de Comandos o resultado do operador lógico NÃO que é a palavra FALSO

Os operadores relacionais: maior (>), menor (<) e igual (=) são usados na construção de expressões lógicas que retornam VERD ou FALSO. Por exemplo:

MOSTRE 738 > 725

O comando MOSTRE imprime na Janela de Comandos o resultado do operador relacional > que é a palavra VERD

MOSTRE 7 = 7.0

O comando MOSTRE imprime na Janela de Comandos o resultado do operador relacional = que é a palavra VERD

MOSTRE "Paulo = [Paulo]

O comando MOSTRE imprime na Janela de Comandos o resultado do operador relacional = que é a palavra FALSO

O operador relacional = possui também uma notação préfixa que é SÃOIGUAIS. Veja mais exemplos:

MOSTRE SÃOIGUAIS " [ ]

O comando MOSTRE imprime na Janela de Comandos o resultado do operador relacional SÃOIGUAIS que é a palavra FALSO

MOSTRE SÃOIGUAIS [ ] [elemento]

O comando MOSTRE imprime na Janela de Comandos o resultado do operador relacional SÃOIGUAIS que é a palavra FALSO

MOSTRE SÃOIGUAIS (PRI POS) COORX

O comando MOSTRE imprime na Janela de Comandos o resultado do operador relacional SÃOIGUAIS comparando o resultado da operação PRI POS e da operação COORX , que é VERD.

Definindo Procedimentos

Os comandos e operações vistos até aqui fazem parte do conjunto de primitivas da linguagem Logo e podem ser usados na definição de novas palavras, isto é, na criação de procedimentos que expandem o conjunto inicial de primitivas da linguagem de programação. A criação de novas palavras é possível graças à atividade de programação.

Embora o mecanismo de definir programas em Logo seja bastante simples, a atividade de programação é extremamente interessante porque obriga o usuário a pensar no processo de solução de um problema e no domínio de conhecimento que será utilizado neste processo.

Além disso é a única maneira de se ter controle sobre o computador, de modo a fazê-lo produzir qualquer resultado que se deseje.

Na maioria das implementações da linguagem Logo, para se definir um procedimento é necessário estar no modo de edição ou no editor de programas.

Suponha que se queira definir um triângulo equilátero com o lado de tamanho 100. A seqüência de comandos a serem dados a Tartaruga poderia ser:

```
PF 100
PD 120
PF 100
PD 120
PF 100
PD 120
```

ou

```
REPITA  3  [ PF  100  PD  120 ]
```

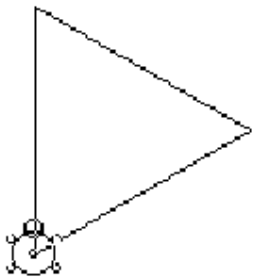
Se desejássemos definir um procedimento Logo que desenhasse este triângulo deveríamos, dentro do modo de edição, teclar:

```
APRENDA TRIÂNGULO
PF  100
PD  120
PF  100
PD  120
PF  100
PD 120
FIM
```

ou

```
APRENDA TRIÂNGULO
REPITA  3  [ PF 100   PD  120 ]
FIM
```

Ao sairmos do modo de edição e retornarmos ao modo direto de uso podemos dar o comando TRIÂNGULO para a Tartaruga, obtendo-se o resultado desejado, como pode ser visto na figura:



Neste caso, dizemos que TRIÂNGULO passa a fazer parte do elenco de comandos que a Tartaruga conhece assim como: UB, TAT etc..

Todo procedimento em Logo tem:

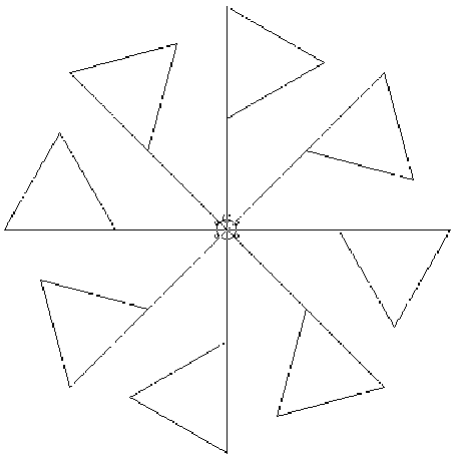
1. Uma linha título que consiste no uso do comando **APRENDA** seguido de um nome[6] que referencia o procedimento
2. Um corpo, representado pelo conjunto de comandos que executarão aquele procedimento
3. Uma linha de finalização que consiste no uso do comando **FIM**

Portanto, um procedimento em Logo é uma seqüência finita de comandos que se caracteriza por duas coisas: o uso do comando APRENDA seguido de um nome e o uso do comando FIM que assinala o término das instruções pertencentes ao procedimento.

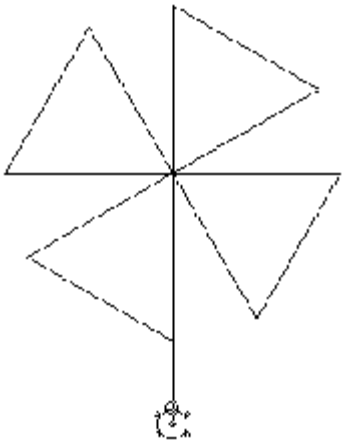
Existe em Logo o conceito de área de trabalho que é uma região de memória do computador onde todos os procedimentos definidos durante uma sessão de trabalho ficam armazenados. No exemplo do TRIÂNGULO, ao sairmos do modo de edição, o procedimento que define o triângulo fica armazenado na área de trabalho enquanto se está usando o Logo. Se desejarmos um armazenamento permanente em disquete ou disco rígido isto deverá ser feito usando-se comandos para manipulação de arquivos específicos de cada implementação.[7]

Estruturando um Projeto Simples

Uma vez definido um procedimento ele pode ser usado na definição de outros procedimentos. Por exemplo, o procedimento TRIÂNGULO visto anteriormente pode ser reutilizado nos seguintes contextos:



```
APRENDA CATAVENTO
REPITA 8 [  PF 100 TRIÂNGULO  PT  100  PD  45  ]
FIM
```



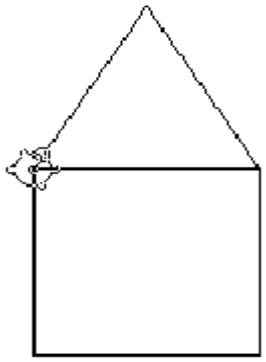
```
APRENDA TREVO
REPITA 4 [ TRIÂNGULO PD 90 ]
PT 150
FIM
```

6 O nome de um procedimento pode, a princípio, ser qualquer palavra mas, uma vez definido o procedimento, ele passa a ser mais uma palavra do repertório do Logo cuja escrita precisa ser rigorosamente respeitada.

7 Vide manual de referência sobre gerenciamento da área de trabalho na implementação do SuperLogo.

Quando um procedimento é usado como comando de um outro procedimento, diz-se que o primeiro é um **subprocedimento** do segundo. No exemplo, TRIÂNGUL0 é um subprocedimento de TREVO.

Suponhamos que se queira implementar procedimentos que desenhem uma casa como a da figura a seguir:



À princípio, pode-se criar um único procedimento que descreve passo-a-passo as ações da Tartaruga. Poderíamos, então, definir o seguinte procedimento:

```
APRENDA CASA
PF  100 PD  90
PF  100 PD  90
PF  100 PD  90
PF  100 PD  90
PF  100 PD  30
PF  100 PD  120
PF  100
FIM
```

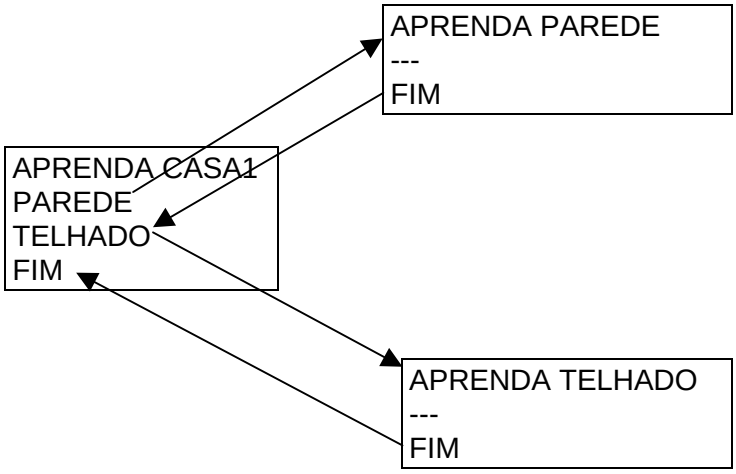
Apesar de ser uma solução, esta forma de definir nem sempre é a mais adequada. No desenvolvimento de projetos mais complexos acaba-se tendo procedimentos excessivamente longos e de difícil entendimento. Imagine o trabalho do usuário para descobrir um giro da Tartaruga equivocado no meio de um procedimento de 50 linhas! Daí a necessidade de introduzirmos a noção de estruturação de procedimentos.

Vejamos então, duas maneiras diferentes de estruturar o projeto **casa**:

APRENDA CASA1

```
PAREDE
TELHADO
FIM

APRENDA PAREDE
REPITA 5 [ PF 100 PD 90 ]
FIM

APRENDA TELHADO
REPITA 3 [  PF 100 PE 120 ]
FIM
```

Esquematicamente, o interpretador Logo, funcionaria da seguinte maneira, durante o processo de execução do procedimento CASA1:



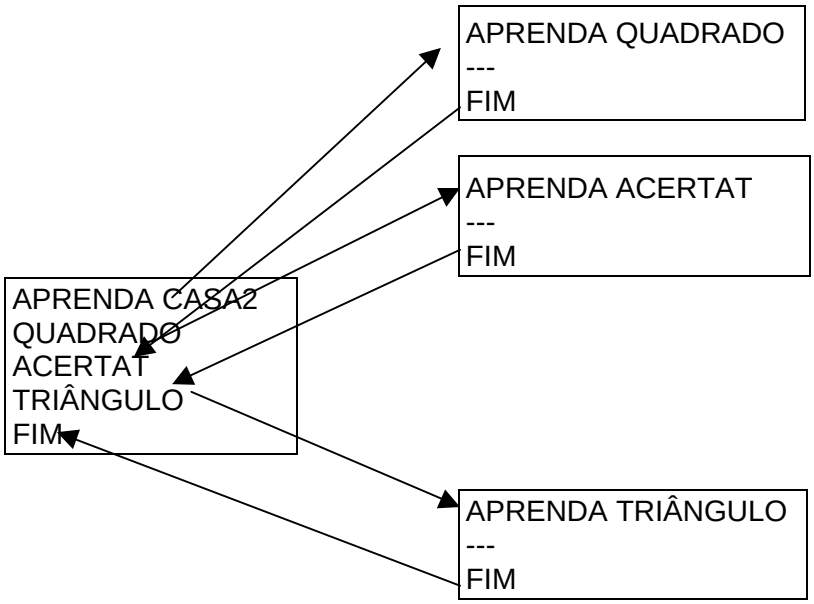
O subprocedimento PAREDE é uma variação de um quadrado. O número de repetições usado pelo comando REPITA deixa a Tartaruga em uma posição ideal, já adequada para a execução do subprocedimento seguinte que é TELHADO. Este, por sua vez, usa um giro para a esquerda que possibilita o acerto do TELHADO sobre a PAREDE. Note que essa solução implica a antecipação da posição e da direção da Tartaruga ao término de um subprocedimento e início do seguinte.

Outra forma de estruturar o projeto **casa** seria assim:

```
APRENDA CASA2
QUADRADO
ACERTATAT
TRIÂNGULO
FIM

APRENDA QUADRADO
```

```
REPITA 4 [PF 100  PD 90 ]
FIM


APRENDA  ACERTATAT
PF 100
PD 30
FIM

APRENDA TRIÂNGULO
REPITA 3 [ PF 100  PD 120 ]
FIM
```

Veja a representação esquemática da execução de CASA2 pelo interpretador Logo:

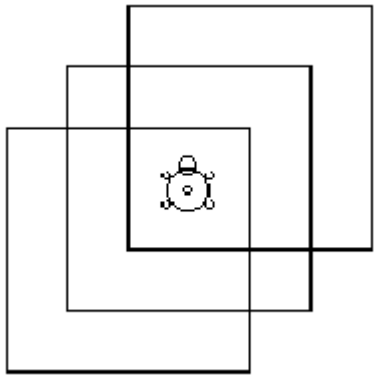


Neste caso, optou-se por um subprocedimento intermediário - ACERTATAT - que faz o acerto da posição e direção da Tartaruga possibilitando o arranjo entre as duas figuras básicas que compõem o desenho original: QUADRADO e TRIÂNGULO.

Os procedimentos QUADRADO e TRIÂNGULO possuem duas características bastante importantes do Logo: a modularidade e a transparência. Diz-se que um procedimento é modular quando ele é independente do seu contexto de uso e, portanto, pode ser reutilizado em muitas outras situações. Por exemplo, o QUADRADO poderia ser usado para fazer uma composição de quadrados como essa:

```
APRENDA COMPOSIÇÃO
REPITA 3 [QUADRADO UN PF 25 PD 90 PF 25 UL PE 90]
FIM
```

Procedimentos como esse facilitam a atividade de programação. O próprio usuário vai criando um arsenal de ferramentas que ele já sabe como funciona.

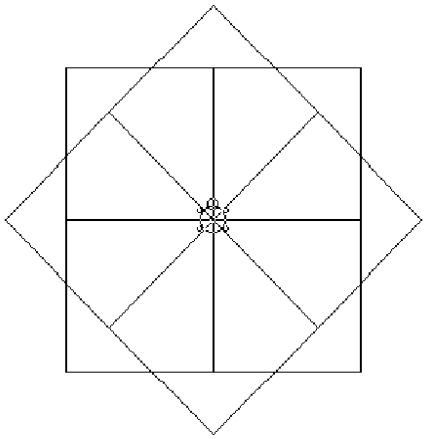


A transparência de um procedimento é parte da modularidade no sentido de que ela torna os procedimentos mais fáceis de serem manipulados. Um procedimento gráfico transparente faz com que a Tartaruga comece e termine o desenho na mesma posição e direção. Dessa maneira, o usuário não precisa se preocupar com a última posição ou direção da Tartaruga para idealizar o procedimento seguinte. Se ele dispõe de peças modulares e transparentes ele passa a se ocupar dos procedimentos de ligação, análogos ao procedimento ACERTATAT do exemplo anterior.

Entretanto, modularidade e transparência não se restringem à direção e posição da Tartaruga. É preciso compreender essas características de forma mais abrangente. Ao realizar um projeto computacional o usuário precisa pensar sobre o estado inicial dos vários objetos que ele estiver manipulando: cor do fundo da tela, cor do lápis, arranjo inicial de posição e orientação, tartaruga com ou sem lápis, aparente ou não etc...

A DIVERSIDADE DE DESCRIÇÕES E A DIVERSIDADE DA ESTRUTURAÇÃO DE PROCEDIMENTOS

Diferentes descrições dos problemas permitem diversas formas de estruturar os procedimentos que solucionam os mesmos. Observemos a figura a seguir e algumas soluções possíveis:



Descrição 1

A Tartaruga desenha dois quadrados sobrepostos. Há um giro entre eles para dispor os quadrados. Em seguida, a Tartaruga desenha um feixe de semi-retas

```
APRENDA QUADRADO
REPITA 4 [ PF 100 PD 90 ]
FIM

APRENDA FEIXE
REPITA 8 [ PF 50 PT 50 PD 45 ]
FIM

APRENDA QUADRADOS1
QUADRADO
PF 50 PD 90
UN PT 20 PE 45 UL
QUADRADO
UN PD 45 PF 70 UL
FEIXE
FIM
```

Descrição 2

A Tartaruga desenha um feixe de semi-retas e, depois dois quadrados sobrepostos. Mantendo um giro entre eles.

```
APRENDA FEIXE
REPITA 8 [ PF 50 PT 50 PD 45 ]
FIM

APRENDA QUADRADO
REPITA 4 [PF 100 PD 90 ]
FIM

APRENDA QUADRADOS2
FEIXE
PT 50 PE 90 PT 50
QUADRADO
PF 50 PD 90
UN PT 20 PE 45 UL
QUADRADO
FIM
```

Descrição3

A Tartaruga desenha um quadrado pequeno e vira um pouco. No total ela desenha 8 quadrados e, portanto o giro entre casa quadrado é de 45 graus.

```
APRENDA QUAPEQUENO
REPITA 4 [ PF 50 PD 90 ]
FIM

APRENDA QUADRADOS3
REPITA 8 [ QUAPEQUENO PD 45 ]
FIM
```

A leitura dos procedimentos nos permite observar a diversidade de descrições na implementação de um mesmo projeto. Selecionamos apenas algumas das descrições possíveis neste contexto. Nosso objetivo não é apontar “a melhor descrição”. Ao contrário, queremos salientar a maleabilidade da atividade de programação que permite a cada pessoa utilizar, criar e descobrir suas próprias estratégias. A atividade de programação, assim compreendida, colabora no sentido de evitar o estabelecimento de soluções padronizadas. Dessa forma, aquele que está aprendendo a programar pode

adquirir flexibilidade tanto na tarefa de resolução de problemas quanto no conhecimento dos recursos oferecidos pela linguagem computacional.

CONSIDERAÇÕES FINAIS

Nosso intuito ao desenvolver este material foi o de organizar informações computacionais relevantes para o leitor que deseja aprofundar e/ou formalizar seus conhecimentos sobre a linguagem Logo do ponto de vista computacional. Não se trata, portanto, de um manual organizador de "aulas" de um curso de Logo. Seu uso independe de um curso e vice-versa. Muitas vezes temos usado este texto como um material de apoio durante os cursos; outras, somente o recomendamos após a experiência do curso. Em outras palavras, sua leitura depende do contexto e do objetivo de uso da linguagem Logo. Note-se, por exemplo, que a definição de procedimentos é iniciada quase no final deste material. Essa escolha restringe-se a um critério de organização de um material escrito e não à abordagem utilizada em laboratório durante a realização de cursos.

BIBLIOGRAFIA

ABELSON, N. e ABELSON, A. *Logo for the Macintosh: An Introduction Through Object Logo.* Paradigm Software Inc, Cambridge, MA, 1992.

Manual do SuperLogo, NIED, UNICAMP, 1994.

PC Logo for Windows: Tutorial, Reference and Glossary. Harvard Associados Inc, 1994.