AutoLISP - II
Cálculos, Variáveis e Listas

João Manuel R. S. Tavares
Joaquim Oliveira Fonseca

Universidade do Porto
Faculdade de Engenharia
FEUP
DEMec
Departamento de Engenharia Mecânica

Cálculos

- Expressões matemáticas são escritas na notação infixa; isto é, o operador antecede os operandos:

1 + 2	⟶	(+ 1 2)
32.12 - 22.5	⟶	(- 32.12 22.5)
1 + 2 + 3 + 4	⟶	(+ 1 2 3 4)
7 * (9 / 2.0)	⟶	(* 7 (/ 9 2.0))

Variáveis:

- A variável é um recurso utilizado pelo programa para executar funções iguais mas com valores que podem ser alterados em cada execução.
- Existem três tipos:
  - livre (global);
  - associada (argumento ou parâmetro);
  - local.
- O tipo da variável é determinado pela sua posição na lista de parâmetros, na definição da função.

Variáveis - tipos (esquema)

```
(defun func1(A B / C D)
    ; A, B: variáveis associadas (parâmetros), C, D: variáveis
                    locais
    ...
    (setq X ...)    ; X variável livre/global
    ...
)
(defun func2()
    (setq Y (+ X 2))      ; Y e X variáveis livres/globais
    ...
)
```

Variáveis - tipos (definições)

- Variável livre:
  - Equivalente a variáveis globais. São utilizadas por várias funções. Não deve fazer parte da lista de parâmetros na definição da função, pois não pertence a nenhuma função específica e sim a todas as funções que constituem o programa.

- Variável local:
  - Equivalente a uma variável temporária. Utilizada para armazenar dados auxiliares dentro de uma determinada função. O seu valor é perdido na saída da função.

Variáveis - tipos (definições)

- Variável associada:
  - Equivalente a parâmetros de entrada de uma função.

- Variáveis associadas e locais após a saída da função assumem o valor anterior à chamada da mesma (por exemplo, o valor zero).

Exemplo - função com variáveis locais

```
;;;Cálculo do perímetro de uma circunferência a partir do
;;;diâmetro inserido pelo utilizador
(defun c:circun(/ DIA PER)
    (setq DIA (getreal "\nDigite o valor do diâmetro: "))
    ;pi é uma constante do programa (Autocad / Autolisp)
    (setq PER (* DIA pi))
    (princ "\nDiametro = ") (princ DIA)
    (princ "\nPerímetro = ") (princ PER)
    ;este princ faz com que a função não retorne um nil
    (princ)
)
```

Exemplo - função com variável associada

- arquivo *progcir.lsp*:

(Cálculo do perímetro de uma circunferência a partir do diâmetro passado como argumento.)

```
(defun c:circun(DIA)
   (setq PER (* DIA pi))
   (princ "\nDiametro = ") (princ DIA)
   (princ "   ")
   (princ “Perimetro = ") (princ PER)
   (princ)
)
```

(DIA – Variável associada)

Exemplo - função com variável associada

- linha de comando do *AutoCAD*:

```
Command: (load "progcir.lsp")
CIRCUN
Command: (c:circun 34)
Diametro = 34   Perimetro = 106.814
Command:
```

(c:xxx) Obrigatório, apesar de definida como comando, devido a conter argumentos.

Exemplo - função com 2 variáveis associadas

```
(defun c:teste()     ; programa principal
  (princ "teste")
  (setq a (getreal "\na?"))
  (setq b (getreal "\nb?"))
  (setq res 0)
  (setq a1 0)
  (setq a2 0)
  (setq b1 0)
  (setq b2 0)
  (soma a b)         ; cham. função soma
  (prompt "\na+b=")
  (prompt (rtos res))
  (prompt "\na1=")
  (prompt (rtos a1))
  (prompt "\na2=")
  (prompt (rtos a2))
  (prompt "\nb1=")
  (prompt (rtos b1))
  (prompt "\nb2=")
  (prompt (rtos b2))
)
(defun soma(n1 n2)    ; função soma
  (setq res (+ n1 n2))
  (setq a1 (* 2 n1))
  (setq a2 (* 4 n1))
  (setq b1 (* 2 n2))
  (setq b2 (* 4 n2))
)
```

Ex. Resultado:

```
teste
a?1
b?2
a+b=3
a1=2
a2=4
b1=4
b2=8
Command:
```

Exemplo - função com variáveis livres

- Conteúdo do arquivo *media.lsp*:

```
;;;Programa principal
(defun c:mediatestes()
    (setq t1 (getreal "\nNota teste-1: "))
    (setq t2 (getreal "\nNota teste-2: "))
    (setq nota (soma-e-divide))
    (princ "\nMedia dos testes = ")
    (princ nota) (princ)
)
;;;funcao soma-e-divide
(defun soma-e-divide()
    (/ (+ t1 t2) 2.0)
)
```

Exemplo - função com variáveis livres

- Carregar o arquivo e executar a função *mediatestes*:

```
Command: (load "c:\\trab\\media.lsp")
SOMA-E-DIVIDE
Command: mediatestes
Nota teste-1: 15.0
Nota teste-2: 17.0
Media dos testes = 16.0
Command:
```

Manipulação de listas de dados

- Dados agrupados em listas são manipulados por funções específicas.
- Existem apenas funções para obter:
  - o primeiro átomo da lista - **função *car***;
  - o resto da lista (todos elementos exceto o primeiro) - **função *cdr***.

Manipulação de listas de dados

- Para obter o 2º, o 3º átomo da lista utilizam-se funções derivadas a partir de concatenações (máximo de 4) de *car* e *cdr*; exemplos:
  - ***cadr*** **-** retorna o 1º átomo do resto da lista original. Portanto, retorna o 2º átomo da lista original.
  - ***cdaddr*** = cdr + car + cdr + cdr
- Para ler uma função derivada de concatenações de *car* e *cdr*:
  1. ler de trás para frente,
  2. desprezar a primeira e a última letra,
  3. um **d** representa um *cdr* e um **a** representa um *car*.

Comandos *car*, *cdr* e derivados

Command: (setq TESTE (list 10 20 30 40))
(10 20 30 40)

Command: !TESTE ← Escreve o valor da variável TESTE
(10 20 30 40)

Command: (car TESTE)
retorna o primeiro átomo da lista
10

Comando *car*, *cdr* e derivados

...
Command: (cdr TESTE)
retorna uma lista sem o primeiro átomo da lista original
(20 30 40)
Command: (cadr TESTE)
executa um cdr sobre a lista original e um car sobre o resultado
20
Command: (caddr TESTE)
executa um cdr sobre a lista original, mais um cdr e finalmente um car sobre os respetivos resultados
30