Ficha de Laboratório Nº 5: Lambda, Let, Meta-funções e Avaliador

Inteligência Artificial - Escola Superior de Tecnologia de Setúbal

Prof. Joaquim Filipe
Eng. Filipe Mariano

Objetivos da Ficha

- Reforçar a aprendizagem da linguagem Lisp
- Funções anónimas e meta-funções
- Definição de variáveis
- Introdução ao avaliador

1. Exemplo de Utilização de Meta-funções em Lisp

1. Descarregue o ficheiro disponível no Moodle, chamado **laboratorio5.lisp** e guarde-o no seu computador.
2. Abra o ficheiro no IDE LispWorks.
3. Observe a estrutura de comentários no inicio do ficheiro para indicar o conteúdo do ficheiro e o seu autor.
4. No ficheiro de suporte ao laboratório, vamos definir a função `remover-se`. A função `remover-se (<predicado> <lista>)` permite reconstruir uma lista sem os elementos que verificam o predicado passado como argumento.
5. Analise os valores dos casos de teste definidos para a função.
6. Inserir o código da função desenvolvida com o docente do laboratório no *Listener*.
7. No *Listener*, testar a função `remover-se` com `pred(x) ≡ (x = 0)`. O comportamento esperado será o retorno a lista sem os átomos `0`, isto é:

```
CL-USER > (remover-se #'(lambda (x) (= x 0)) '(1 2 0 2 0 4))
(1 2 2 4)
```

8. Copie o código da função que foi testada para o ficheiro, de modo a armazenar a mesma.

2. Exercícios Introdutórios

2.1. Parâmetros de Lambda

1. `media`: Função que retorna a média de um conjunto variável de valores numéricos passados como argumento. Usar o parâmetro `&rest` para criar a função. Testar a função com os valores `1 2 34` e `7 8 10 3.5`.

```
CL-USER > (media 1 2 34)
12.333333

CL-USER > (media 7 8 10 3.5)
7.125
```

2. `coluna`: Permite obter a primeira coluna de uma matriz quadrada. Re-escreva a função de forma a substituir a instrução `let` pelo parâmetro `&aux` da função Lambda.

```
CL-USER > (defun coluna (m1) (mapcar #'(lambda (linha) (let ((cabeca (car
linha))) cabeca)) m1))
COLUNA

CL-USER > (coluna '((1 2 3)(4 5 6)(7 8 9)))
(1 4 7)
```

2.2. Avaliador

3. `aplicar-simbolo`: Retorna a avaliação da instrução representada pelo símbolo, a um par de valores numéricos passados como argumento. Usar a instrução `eval` para criar a função e testar a mesma com os símbolos `mod` e `*` e para os valores `2` e `3`.

```
CL-USER > (aplicar-simbolo 'mod 2 3)
2

CL-USER > (aplicar-simbolo '* 2 3)
6
```

Tabela 1: Lista dos alunos da Turma 1.

Nome	Apelido	Notas
Joao	Silva	12.5 15 8.5
Ana	Santos	11.5 18 13.5
Paulo	Jorge	6.5 10 7.5
Elisabete	Navarro	12.5 15 8.5
Mario	Rodrigues	12.5 15 8.5

4. `turma-1`: Retorna uma lista com informação sobre os alunos de uma turma, valores estes ilustrados na Tabela 1. Cada aluno tem um nome, um apelido e um conjunto de três notas. Testar a função no *Listener*.

```
    CL-USER > (turma-1)
    (("Joao" "Silva" (12.5 15 8.5)) ("Ana" "Santos" (11.5 18 13.5)) ("Paulo"
"Jorge" (6.5 10 7.5)) ("Elisabete" "Navarro" (12.5 15 8.5)) ("Mario" "Rodrigues"
(12.5 15 8.5)))
```

5. `nome`: Retorna o nome de um aluno. Testar a função com o primeiro aluno da `turma-1`.

```
    CL-USER > (nome (first (turma-1)))
    "Joao"
```

6. `apelido`: Retorna o apelido de um aluno. Testar a função com o primeiro aluno da `turma-1`.

```
    CL-USER > (apelido (first (turma-1)))
    "Silva"
```

7. `notas`: Retorna a lista de notas de um aluno. Testar a função com o primeiro aluno da `turma-1`.

```
    CL-USER > (notas (first (turma-1)))
    (12.5 15 8.5)
```

2.3. Definição de Variáveis e Meta-funções

8. `media-das-notas`: Retorna o valor da média das notas contidas numa lista de valores numéricos passada como argumento. Usar a instrução `let` para armazenar localmente o numero de notas. Usar a instrução `apply` para construir a função. Testar a função com a lista `10 15 20`.

```
    CL-USER > (media-das-notas '(10 15 20))
    15
```

9. `media-da-turma`: Retorna a média das notas obtidas pelos alunos de uma turma, representada por uma lista de alunos passada como argumento. A função utiliza `media-das-notas` (definida no exercício anterior), para calcular a média das notas de um aluno. Usar a instrução `let` para armazenar localmente o número de alunos da turma. Usar uma função `lambda` e as instruções `mapcar` e `apply` para construir a função. Testar a função com a lista `turma-1`.

```

```

```
CL-USER > (media-da-turma #'media-das-notas (turma-1))
11.666666
```

10. `percentagem-aprovados`: Retorna a percentagem de alunos aprovados numa turma. Esta turma é representada por uma lista de alunos passada como argumento. Usar a instrução `let` para armazenar localmente o número de alunos da turma. Usar as instruções `lambda`, `mapcar` e `apply` para construir a função. Testar a função com a lista `turma-1`.

```
CL-USER > (percentagem-aprovados (turma-1))
80.0
```

11. `lista-dos-aprovados`: Retorna a lista de alunos aprovados numa turma. Esta turma é representada por uma lista de alunos passada como argumento. Usar as instruções `lambda` e `mapcar` para construir a função. Testar a função com a lista `turma-1`.

```
CL-USER > (lista-dos-aprovados (turma-1))
(("Joao" "Silva") ("Ana" "Santos") NIL ("Elisabete" "Navarro") ("Mario"
"Rodrigues"))
```

12. `todos-aprovadosp`: Retorna o valor verdadeiro (`T`) se todos os alunos de uma turma tiverem aproveitamento (média superior a `9.5`) e `NIL` caso contrário. Esta turma é representada por uma lista de alunos passada como argumento. Usar as instruções `let*` e `mapcar`, assim como a função `lista-dos-aprovados` e uma função `lambda` para armazenar localmente a lista dos alunos da turma e a lista correspondente com os valores `T` e `NIL`. Usar igualmente as instruções `eval` e `and` para construir a função. Testar a função com a lista `turma-1`.

```
CL-USER > (todos-aprovadosp (turma-1))
NIL
```

13. `avaliar-turma`: Retorna a avaliação de uma turma de alunos, de acordo com uma turma e uma função passadas como argumento. A função deve possibilitar a invocação sem argumentos de entrada. Se for o caso, a função passa a utilizar a função `percentagem-aprovados` por defeito para avaliar a a turma. Utilizar o parâmetro `&optional`. Testar a função sem argumento de entrada para o parâmetro da função e tendo como argumento uma das funções definidas anteriormente.

```
CL-USER > (avaliar-turma (turma-1))
80.0

CL-USER > (avaliar-turma (turma-1) #'lista-dos-aprovados)
(("Joao" "Silva") ("Ana" "Santos") NIL ("Elisabete" "Navarro") ("Mario"
"Rodrigues"))
```

14. `existep`: Retorna o valor verdadeiro (`T`) se encontrar um determinado aluno numa turma. Utilizar as instruções `eval`, `mapcar`, `and` e `or`, assim como a função `lambda`. Testar a função com os valores `"Joao"`, `"Silva"` e `turma1`.

```
CL-USER > (existep "Joao" "Silva" (turma-1))
T
```