LISP
PARADIGMA FUNCIONAL

Pedro Dias

ÍNDICES

PARADIGMA FUNCIONAL

O Paradigma Funcional trata a computação como a avaliação de funções matemáticas, evitando mudanças de estado e dados mutáveis.

Diferença da POO: Sem estado mutável, sem "objetos" interagindo, o foco é na transformação de dados por funções.

CONCEITOS FUNDAMENTAIS

1. Funções Puras: A saída depende apenas dos argumentos de entrada (sem efeitos colaterais).
   Imutabilidade: Dados não mudam depois de criados.

2. Expressões vs. Comandos: Tudo é uma expressão que retorna um valor.
   Recursão: Usada para iteração em vez de loops tradicionais

3. Composição de Funções: Combinar funções menores para construir funções complexas.

A LINGUAGEM LISP

- **Origem: Segunda linguagem de programação de alto nível (depois do Fortran), criada por John McCarthy (1958).**

- **LISP = LISt Processing: Tudo no código e nos dados é representado como listas.**

- **S-Expressions: A sintaxe é uniforme**

IMUTABILIDADE
FUNÇÕES PURAS

Em Lisp, a ênfase é em criar novos dados em vez de modificar os existentes.

Função Pura:
(define (dobro x) (* x 2)) Depende apenas de x

Função com Efeito Colateral (Exemplo a Evitar):
(begin (display x) (* x 2)) - Efeito Colateral: imprimir no console.

COMPOSIÇÃO DE FUNÇÕES

- **Origem: Segunda linguagem de programação de alto nível (depois do Fortran), criada por John McCarthy (1958).**

- **LISP = LISt Processing: Tudo no código e nos dados é representado como listas.**

- **S-Expressions: A sintaxe é uniforme**

TUDO É UMA EXPRESSÃO

A principal característica do Lisp: o código e os dados têm a mesma estrutura (listas).

- **Expressão: (+ 1 2)**
- **Lista de Dados: '(1 2 3) (A aspas ' indica que é uma lista de dados, não código para ser executado.)**

Isso permite que programas escrevam ou manipulem outros programas.

EXERCÍCIO 1: FARMÁCIA DA MAOMAO

Maomao precisa de um sistema simples para determinar a dosagem e o preço de ervas e medicamentos.

1. Função Pura: calcula-dosagem Crie uma função pura (calcula-dosagem peso-kg idade-anos) que retorna a dosagem base (ml) seguindo estas regras:

- **Se a idade for menor que 5 ou o peso for menor que 20kg, a dosagem é 10ml.**
- **Se a idade for entre 5 e 12 (inclusive) e o peso for maior ou igual a 20kg, a dosagem é 25ml.**
- **Caso contrário, a dosagem é 50ml.**

2. Função Pura: ajusta-preco Crie uma função pura (ajusta-preco preco-base nome-da-erva) que aplica um fator de preço com base no nome:

- **Se o nome for "Ginseng", multiplica o preço por 3.0.**
- **Se o nome for "Lótus", multiplica o preço por 1.5.**
- **Caso contrário, retorna o preco-base.**

3. Calcule o preço final de uma dose de "Lótus" (preço base 10 moedas) para um paciente de 14 anos e 60kg. (teste outras para mostrar os outros casos)

EXERCÍCIO 2: CATÁLOGO DE FAUNA

Crie um sistema para gerenciar o catálogo de fauna de Planet 4546B.

1. Definição da Estrutura: Defina uma estrutura chamada criatura com os campos: (nome, ambiente, periculosidade, vida-media).

2. Catálogo Inicial: Crie uma lista (catálogo) com pelo menos 4 criaturas, incluindo:

- **Um peixe Safe Shallows (Periculosidade Baixa).**
- **Um Reaper Leviathan (Periculosidade Alta).**
- **Duas criaturas quaisquer com ambiente Deep.**

3. HOF: Filtro de Perigo Crie uma função (filtra-por-perigo catalogo) que utiliza filter (ou remove-if-not) para retornar apenas as criaturas cuja periculosidade não seja Baixa.

4. Mapeamento de Informações: Crie uma função (relatorio-profundidade catalogo) que utiliza mapcar para retornar uma lista de strings no formato: "[NOME]: Vive em [AMBIENTE]", mas apenas para criaturas do ambiente "Deep". Dica: Você pode usar o filtro antes de mapcar (crie casos para testar)

EXERCÍCIO 3: A LOJA DE IWAI

Você é o gerente de uma loja clandestina que vende itens amaldiçoados e artefatos mágicos.

1. Estrutura e Catálogo: Defina uma estrutura item com campos (nome, tipo, preco, forca-magica). Crie uma lista de itens, incluindo armas (Arma), poções (Pocao) e artefatos (Artefato).

2. Funções de Transformação:

- Crie uma função pura (adiciona-imposto preco) que aumenta o preço em 15%.
- Crie uma função pura (bonus-maldicao forca) que retorna forca * 1.5 se a forca-magica for maior que 80.

3. Crie uma função (processa-venda catalogo) que executa a seguinte ordem a. Filtra: Mantém apenas os itens do tipo Arma. b. Transforma (1): Utiliza mapcar e um lambda para aumentar o preço de cada arma usando adiciona-imposto. c. Transforma (2): Utiliza mapcar para retornar o nome da arma e seu novo valor de forca-magica (aplicando bonus-maldicao na forca-magica original do item).

EXERCÍCIO 4:
GERENCIAMENTO DE OCORRÊNCIAS

Você deve criar um sistema para analisar o "Nível de Exposição Paranormal" de missões da Ordem.

1. Estrutura: Defina uma estrutura ocorrencia com campos: (nome, ritual, nivel-medo, agentes-enviados).

2. Função Recursiva: soma-medo-recursiva Crie uma função recursiva (soma-medo-recursiva lista-ocorrencias) que percorre a lista e retorna a soma total do campo nivel-medo de todas as ocorrências.

3. Função de Alto Nível (Composição e Local Scope): Crie uma função (analise-final lista-ocorrencias) que realiza o seguinte: a. Calcula Média: Usa a função recursiva soma-medo-recursiva e length (ou list-length) para calcular a média do nivel-medo e armazena em uma variável local (let). b. Filtra Ocorrências Críticas: Utiliza filter e um lambda para retornar uma lista com os nomes das ocorrências que têm mais de 3 agentes-enviados e um nivel-medo acima da média calculada no passo (a). c. Composição: O resultado final da função deve ser apenas a lista de nomes das ocorrências críticas.

Teste Final: Crie uma lista com 4 ou 5 ocorrências e chame a função (analise-final lista-ocorrencias).

Paradigma Funcional| LISP

FIM DA AULA

GitHub:
https://github.com/zSh3rl0cK/S01-monitoria