Linguagens de Programação Funcional

Elaboração: Prof. Raimundo Barreto
Revisão: Prof. Alberto Castro
DCC/ICE/UA

Sumário

- Introdução
- Fundamentos de Linguagens de Programação Funcionais
- LISP
- Aplicações
- Comparação entre Linguagens Funcionais e Imperativas

Introdução

- É uma categoria de linguagens não-imperativas
- Imperativas: uso eficiente das arquiteturas de computadores de Von Neumann
- A arquitetura não deveria ser uma restrição no processo de desenvolvimento de software
- Alguns outros paradigmas existem. Não eficientes. Não dominaram o mercado

Introdução

- O paradigma de programação funcional é baseado em funções matemáticas
- LISP iniciou como uma linguagem funcional pura, mas adquiriu algumas características imperativas
- Outras linguagens: Scheme, Common Lisp, ML, Miranda, etc.

Fundamentos

- Uma função matemática é um mapeamento de membros de um conjunto (domínio) em outro conjunto (contra-domínio)
- Uma definição de função especifica os dois conjuntos e a forma de mapeamento
- O mapeamento é descrito por uma expressão
- Funções podem ser aplicadas a um elemento particular do domínio

Fundamentos

- O domínio pode ser um produto cartesiano entre vários conjuntos
- Uma função sempre retorna um valor do contra-domínio
- Ordem de avaliação das expressões de mapeamento são controladas por recursão e condicionais, ao invés de seqüências e repetições
- Dados os mesmos argumentos sempre produz o mesmo resultado. Não produz efeito colateral

Fundamentos

- Definição de função: nome de função, lista de parâmetros e expressão de mapeamento

- Funções complexas são definidas em termos de outras funções.
- Forma funcional é uma que recebe funções como parâmetros ou retorna uma função como resultado
- Forma comum de forma funcional: composição funcional

Fundamentos

- Composição funcional:
  - Tem duas funções como parâmetros
  - Retorna uma função cujo resultado é a primeira função aplicada no resultado da segunda
  - É expressa da seguinte forma:

$h \equiv f \circ g$

por exemplo: se
$f(x) = x+2$
$g(x) = 3*x$

então:
$h(x) = f(g(x)) = (3*x)+2$

Fundamentos

- Objetivo de projeto: imitar funções matemáticas
- Abordagem de solução de problemas diferente de métodos usados em programação imperativa
- Imperativa: célula de memória e variáveis
- Funcional: não existe variáveis ou atribuição
- Livra o programador de lidar com memória
- Repetição é feita via recursão

Fundamentos

- Programas são definições e aplicações de funções
- Execução: avaliação das aplicações das funções
- Semântica mais simples
- Linguagens imperativas provêem algum suporte para programação funcional.
- Alguns problemas:
  - Restrição no tipo de valor que pode retornar
  - Podem causar efeitos colaterais

Fundamentos

- Uma linguagem funcional provê:
  - conjunto de funções primitivas
  - conjunto de formas funcionais para construir funções complexas a partir das funções primitivas
  - operação de aplicação de funções
  - alguma forma de armazenamento de dados

LISP

- LISP: A mais antiga e mais amplamente usada
- Com exceção da primeira versão, todos os dialetos incluem algumas características de linguagens imperativas, tais como:
  - Variáveis
  - Instruções de atribuição
  - Iteração

LISP

Tipos de Dados e Estruturas

- LISP é uma linguagem sem tipos
- Dois tipos de objetos de dados: átomos e listas
- Átomos são os símbolos de LISP
- Constantes numéricas são também átomos
- Estrutura de dados: listas

LISP

Tipos de Dados e Estruturas

- Listas são especificadas delimitando seus elementos por parêntesis

(A B C D)

- Estrutura de listas aninhadas é também possível

(A (B C) D (E (F G)))

LISP

O primeiro interpretador LISP

- Notação-M (de Meta-notation): A primeira notação de LISP. Mais próxima possível de FORTRAN.
- McCarthy acreditava que processamento de listas poderia substituir as máquinas de Turing no estudo de computabilidade
- Construção de uma função LISP universal que pudesse avaliar qualquer outra função em LISP

LISP

O primeiro interpretador LISP

- Uma notação que permitisse que funções fossem expressas da mesma forma que dados

(nome_função argumento_1 ... argumento_n)

Ex.: (+ 5 7)

- Definição de funções: notação lambda

(nome_função (LAMBDA(arg_1...arg_n) expressão))

LISP

O primeiro interpretador LISP

- EVAL: uma função universal que avaliava outras funções
- EVAL tornou-se um interpretador LISP: implementação rápida, fácil e não esperada
- A notação-M nunca foi implementada.
- S-expressions: expressões simbólicas. Dados e código

Aplicações

- LISP é versátil e poderosa
- LISP foi desenvolvida para computação simbólica e aplicações de processamento de listas
- Editor de texto escrito em LISP: EMACS
- Cálculos simbólicos: MACSYMA
- Construção de sistemas experimentais
- Ensino introdutório de programação

Aplicações

- LISP na Inteligência Artificial:
  - Sistemas especialistas
  - Representação do conhecimento
  - Aprendizado de máquina
  - Processamento de linguagem natural
  - Sistema de treinamento inteligente
  - Modelagem da fala e visão

Comparação entre Funcional e Imperativo

- **Linguagem imperativa**:
  - gerência de variáveis e atribuição de valores
  - eficiência
  - construção de programas requer mais tempo e esforço
- **Linguagem Funcional**:
  - Não se preocupa com variáveis
  - ineficiência
  - programação de altíssimo nível. Não requer muito tempo e esforço (principal vantagem)
- **Programação Funcional**:
  - Sintaxe muito simples: estrutura de listas
  - Semântica também simples: tudo são funções