PARADIGMA FUNCIONAL

- Exemplos de linguagens funcionais
- LISP
  - Início: LP puramente funcional
  - Depois: acréscimo de alguns recursos imperativos (aumentaram eficiência de execução)
  - Ainda é a mais importante das LPs funcionais
- Scheme
  - Dialeto pequeno do Lisp
- COMMON LISP
  - Mistura de diversos dialetos da década de 80 do LISP
- ML
  - Linguagem funcional fortemente tipada
  - Mais sintaxe funcional do que o LISP ou Scheme
- Haskell
  - Baseia-se parcialmente na ML, mas é puramente funcional

PARADIGMA FUNCIONAL: INTRODUÇÃO

- Possui algumas vantagens em relação ao paradigma imperativo

Visualização uniforme dos programas como funções

Tratamento das funções como dados

Limitação do *side effect*

Uso de gerenciamento de memória automático

Grande flexibilidade

Notação concisa

Semântica simples

- Desvantagem: Ineficiência de execução

  Devido a sua natureza dinâmica LP funcionais são interpretadas

  Recentemente: avanços nas técnicas de compilação + arquiteturas especiais

- Adequado para:

  Inteligência artificial

  Sistemas para área da matemática

  Aplicações lógicas

- LP Imperativa X Funções

- Caracterizada por três conceitos: variáveis, atribuições e seqüências - Estado de um programa mantido por variáveis de programa - Variáveis associadas com posições de memória (endereço + valor) - Acesso a variáveis direto ou indireto (através endereços) - Alteração das variáveis através de comandos de atribuição que introduzem dependência no programa	- Em matemática, variáveis são amarradas a valores e não trocam de valor - Função matemática define um mapeamento de um valor do domínio para um valor da imagem - Relaciona cada elemento do domínio com apenas um da imagem
- Resultado de um programa depende da ordem dos comandos de atribuição	- Valor de uma função não depende da ordem de execução
- Laços são usados para processar valores (percorrem seqüências de localização de memória)	- Valores processados através da aplicação de funções; Recursão é usada no lugar da iteração (laço) junto com expressões de condição
- Baseadas em estado e orientada a comandos	- Baseada em valor e aplicativa

- Resumindo
  - Programação Imperativa: variáveis, comandos e procedimentos
  - Programação Funcional: expressões e funções
- Exemplo: cálculo do fatorial (“tradução”)
  - Pascal

```
function factorial (n: integer) : Integer;
    var f : Integer;
    begin
    f := 1;
    while n > 0 do
        begin
        f := f * n;
        n := n - 1
        end;
    factorial := f
    end
```

  - *ML*

```
fun factorialloop (n, f) =
    if n > 0
    then factorialloop (n-1, f*n)
    else f
fun factorial (n) = factorialloop (n, 1)
```

Geralmente "tradução" entre paradigmas não é tão direta

Variáveis globais ≠ parâmetros locais

Bom programa imperativo
≠
Bom programa funcional

Função anterior pode ser melhor escrita “usando o paradigma funcional”

```
fun factorial (n) =
    if n > 0
    then n * factorial (n-1)
    else 1
```

Programa pode ser feito sem usar variáveis locais e laços

PARADIGMA FUNCIONAL: FUNÇÕES MATEMÁTICAS E FUNÇÕES EM LINGUAGEM DE PROGRAMAÇÃO

- Componentes básicos de uma função
  - Domínio: conjunto de objetos aos quais a função pode ser aplicada
  - Imagem: conjunto de objetos que podem resultar da aplicação de uma função
  - Definição: especificação de como um elemento da imagem é determinado a partir de um elemento do domínio
  - Nome
- Paradigma funcional
  - Baseado no conceito matemático de função
  - Mapeamento:Domínio (E) → Imagem (S)

- Definição de uma função
  - Assinatura (especifica o domínio e a imagem)
  - Regra de Mapeamento (especifica o valor da imagem associado com cada valor do domínio)
- Aplicação de uma função
  - Elemento particular do domínio (argumento)
  - Resulta no elemento associado na imagem
- Exemplo: definição da função *double*
  - Domínio: conjunto dos inteiros
  - Imagem: conjunto dos inteiros
  - Definição: x + x (x é um elemento do domínio)
  - Nome: *double*
- Notação matemática:
  - Assinatura, *double: integer->integer*
  - Regra de Mapeamento, $double(x) \equiv x + x$
  - Aplicação, *double(2)*

- Características funções matemáticas
  - Ordem de avaliação das expressões controlada por recursão e expressões condicionais
  - Não possuem efeitos colaterais (definem sempre o mesmo valor, dado o mesmo conjunto de dados de entrada)
  - Define um valor ao invés de especificar uma seqüência de operações sobre variáveis
  - Regra de mapeamento definida em termos de combinações ou aplicações de outras funções
  - Definidas recursivamente
- LP Imperativas
  - Funções definidas de forma seqüencial
  - Mapeamento feito através de passos ordenados
  - Repetição (laço)
  - Variáveis (localização de memória) podem causar efeitos colaterais

PARADIGMA FUNCIONAL: PROGRAMAÇÃO FUNCIONAL

- LP imperativa

  Avaliação da expressão e armazenamento em memória (variável)

  Exemplo: (x + y) / (a - b)

- Objetivos das LP funcionais

  "Imitar" funções matemáticas genericamente

  Nova abordagem

- Características das LP funcionais

Não usa variáveis e comandos de atribuição

Sem variáveis, sem controle de laços

Repetição através da recursão

Programas = definições de funções + especificações de aplicações de funções

Execução

- Mesmos parâmetros, mesmos resultados
- Mecanismos: amarração e aplicação
- Amarração: associa valores com nomes
- Aplicação: processa novos valores

Componentes das LP funcionais

Data Objects (exemplo: lista ou vetor)

Built-in Functions (para manipulação dos objetos de dados)

Functional Forms (funções de alta ordem para construção de novas funções)