PARADIGMAS E LINGUAGENS DE PROGRAMAÇÃO

ENGENHARIA DA COMPUTAÇÃO – UFC/SOBRAL

Prof. Danilo Alves

danilo.alves@alu.ufc.br

INTRODUÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- O projeto das linguagens imperativas é baseado diretamente na arquitetura de computadores von Neumann

  – Eficiência é a principal preocupação, em vez da adequação da linguagem para desenvolvimento de *software*

- Apesar do estilo imperativo ser aceitável pela a maioria dos programadores, sua forte dependência da arquitetura subjacente é vista por alguns como uma **restrição desnecessária** no processo de desenvolvimento de *software*

INTRODUÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- O projeto das linguagens funcionais é **baseado em funções matemáticas**

  – Uma sólida base teórica, também próxima do usuário, mas relativamente despreocupada com a arquitetura das máquinas em que os programas serão executados

ALGUMAS LINGUAGENS FUNCIONAIS

UNIVERSIDADE FEDERAL DO CEARÁ

- LISP começou como uma linguagem puramente funcional, mas incorporou em seguida, recursos imperativos importantes que aumentaram sua eficiência de execução
- Dialetos de LISP
  - Scheme – Utiliza escopo estático
  - COMMON LISP – Mistura de diversos dialetos de LISP
- ML, Miranda e Haskell são outras linguagens cujo paradigma é o funcional

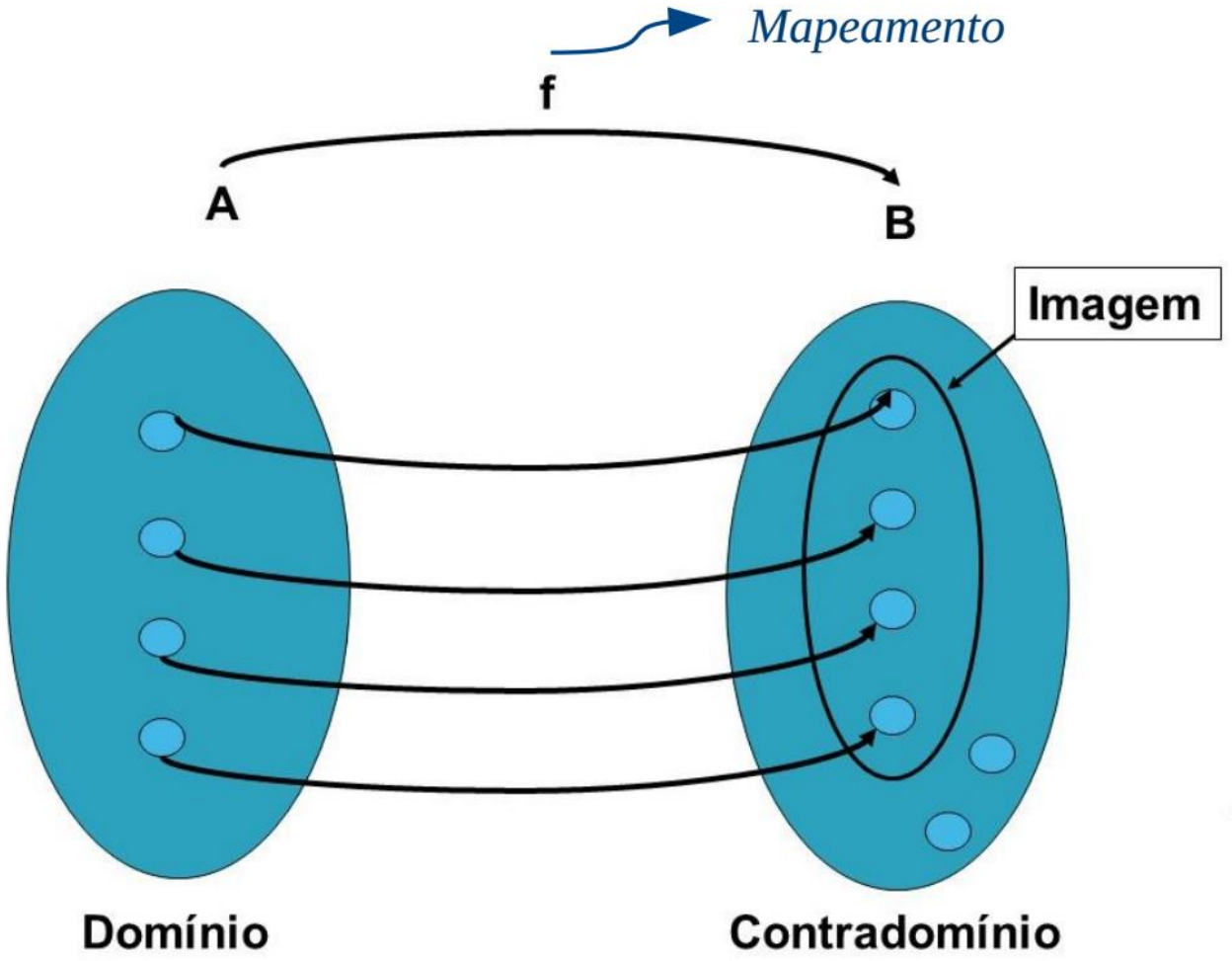
FUNÇÕES MATEMÁTICAS

UNIVERSIDADE FEDERAL DO CEARÁ

- Uma função matemática é um **mapeamento** de membros de um conjunto, chamado de **conjunto domínio**, para outro conjunto, chamado de **conjunto imagem**

  – Uma das características fundamentais das funções matemáticas é que a ordem de avaliação de suas expressões de mapeamento é controlada por **recursão** e **expressões condicionais**, e não por sequência e repetição iterativa, comuns nas linguagens de programação imperativas

FUNÇÕES MATEMÁTICAS



Ordem de avaliação é controlada por ***recursão*** *e por* ***expressões condicionais***

$$f(n) \equiv \begin{cases} 1 \text{ if } n = 0 \\ n * f(n-1) \text{ if } n > 0 \end{cases}$$

Efeitos colaterais não existem

FUNÇÕES MATEMÁTICAS

UNIVERSIDADE FEDERAL DO CEARÁ

- Outra característica importante é que elas não têm **efeitos colaterais**. Sempre definem o mesmo valor quando fornecido o mesmo conjunto de argumentos.

  – Na matemática, não existe algo como uma variável que modela uma posição de memória

  – Variáveis locais em funções, em linguagens de programação imperativas, mantêm o estado da função. Na matemática, não existe o conceito do estado de uma função

  – Uma função matemática define um valor, em vez de especificar uma sequência de operações sobre valores em memória para produzir um valor

  – Não existem variáveis no sentido das variáveis imperativas, então não podem existir efeitos colaterais

FUNÇÕES MATEMÁTICAS

- Definições de funções são geralmente escritas como um nome de função, seguido de uma lista de parâmetros entre parênteses, seguidos pela expressão de mapeamento
- Por exemplo

  cubo(x) ≡ x * x * x, onde x é um número real

  x = 2 → cube (2.0) ≡ 2.0 * 2.0 * 2.0 = 8.0

"é definido como"

Domínio e imagem são os números reais

FUNÇÕES MATEMÁTICAS

UNIVERSIDADE FEDERAL DO CEARÁ

- Uma **expressão lambda** especifica o parâmetro e o mapeamento de uma função com a seguinte forma

  λ(x) x * x * x

  para a função cubo(x) = x * x * x

- Definem uma função anônima

  – Recurso usado por Python e Javascript, adicionado no Java 8

Ex. em Javascript:

```
function cubo(x){
    return x * x * x
}
```

```
cubo = function(x) {
    return x * x * x
}
```

```
cubo = x => x * x * x
```

EXPRESSÕES LAMBDA

UNIVERSIDADE FEDERAL DO CEARÁ

- Expressões lambda descrevem funções sem nome
- São aplicadas aos parâmetros colocando-se os parâmetros depois da expressão

  por exemplo, (λ(x) x * x * x)(2)

  que resulta no valor 8

- Expressões lambda, como outras definições de função, podem ter mais de um parâmetro

FORMAS FUNCIONAIS

UNIVERSIDADE FEDERAL DO CEARÁ

- Uma função de ordem superior, ou **forma funcional**, é uma função que

– recebe funções como parâmetros ou leva a uma função como resultado

- Em linguagens multiparadigmas, como Python e Javascript, torna-se possível atribuir uma função a uma variável

Ex. em Python:

```
soma = lambda a, b: a + b
cubo = lambda x: x*x*x

res = soma(2, 3)  # retorna 5 para res
res = cubo(2)     # retorna 8 para res
```

COMPOSIÇÃO DE FUNÇÕES

UNIVERSIDADE FEDERAL DO CEARÁ

- É uma forma funcional que tem dois parâmetros funcionais ou leva a uma função cujo valor é o primeiro parâmetro de função real aplicado ao resultado do segundo

Forma: $h \equiv f \circ g$
que significa $h(x) \equiv f(g(x))$
Para $f(x) \equiv x + 2$ e $g(x) \equiv 3 * x$,
$h \equiv f \circ g$ resulta $(3 * x) + 2$

Ex. em Python:

```
f1  = lambda x: x+2
f2  = lambda x: 3*x
h   = lambda x, f, g: f(g(x))
res = h(3, f1, f2)  # retorna 11 para res
```

APLICAR-PARA-TODOS

UNIVERSIDADE FEDERAL DO CEARÁ

- É uma forma funcional que recebe uma única função como um parâmetro e produz uma lista de valores obtidos pela aplicação da função de cada elemento de uma lista de parâmetros

Forma: α
Para $h(x) \equiv x * x$
$\alpha(h, (2, 3, 4))$ resulta $(4, 9, 16)$

Ex. em Python:

```
h = lambda x: x*x
a = lambda h, lista: [h(i) for i in lista]
res = a(h, [2, 3, 4])  # retorna [4, 9, 16] para res
```

FUNDAMENTOS DAS LINGUAGENS DE PROGRAMAÇÃO FUNCIONAL

- O objetivo do projeto de uma linguagem de programação funcional **é se aproximar de funções matemáticas** ao máximo possível
- A abordagem para a solução de problemas é fundamentalmente diferente de abordagens usadas com linguagens imperativas

  – Em uma linguagem imperativa, as operações são realizadas e os resultados são armazenados em variáveis para uso posterior

  – Gestão das variáveis é uma preocupação constante e uma fonte de complexidade para a programação imperativa

  – Ex.:

  (x + y) / (a – b)

  - (x+y) é calculada primeiro e tem seu resultado armazenado enquanto (a-b) é avaliada
  - O armazenamento é necessário e é manipulado pelo compilador, embora isso seja **escondido do programador**

FUNDAMENTOS DAS LINGUAGENS DE PROGRAMAÇÃO FUNCIONAL

- Em uma linguagem de programação funcional, variáveis não são necessárias, como é o caso da matemática
- Sem variáveis, as construções de iteração não são possíveis! Pois são controladas por variáveis
  - Repetições devem ser especificadas por meio de recursão
- Os programas são definições de funções e especificações de aplicações de funções, e as execuções consistem em avaliar a aplicação de funções

FUNDAMENTOS DAS LINGUAGENS DE PROGRAMAÇÃO FUNCIONAL

- **Transparência referencial:** Em uma linguagem de programação funcional, a avaliação de uma função sempre produz o mesmo resultado, dados os mesmos parâmetros
- **Recursão em cauda:** Escrever funções recursivas que podem ser convertidas automaticamente para iteração, pelo compilador

  – Apesar de linguagens funcionais serem implementadas com um interpretador, os programas nelas podem ser compilados

  – Uma função é recursiva em cauda se sua chamada recursiva é a última operação na função

TIPOS DE DADOS E ESTRUTURAS LISP

UNIVERSIDADE FEDERAL DO CEARÁ

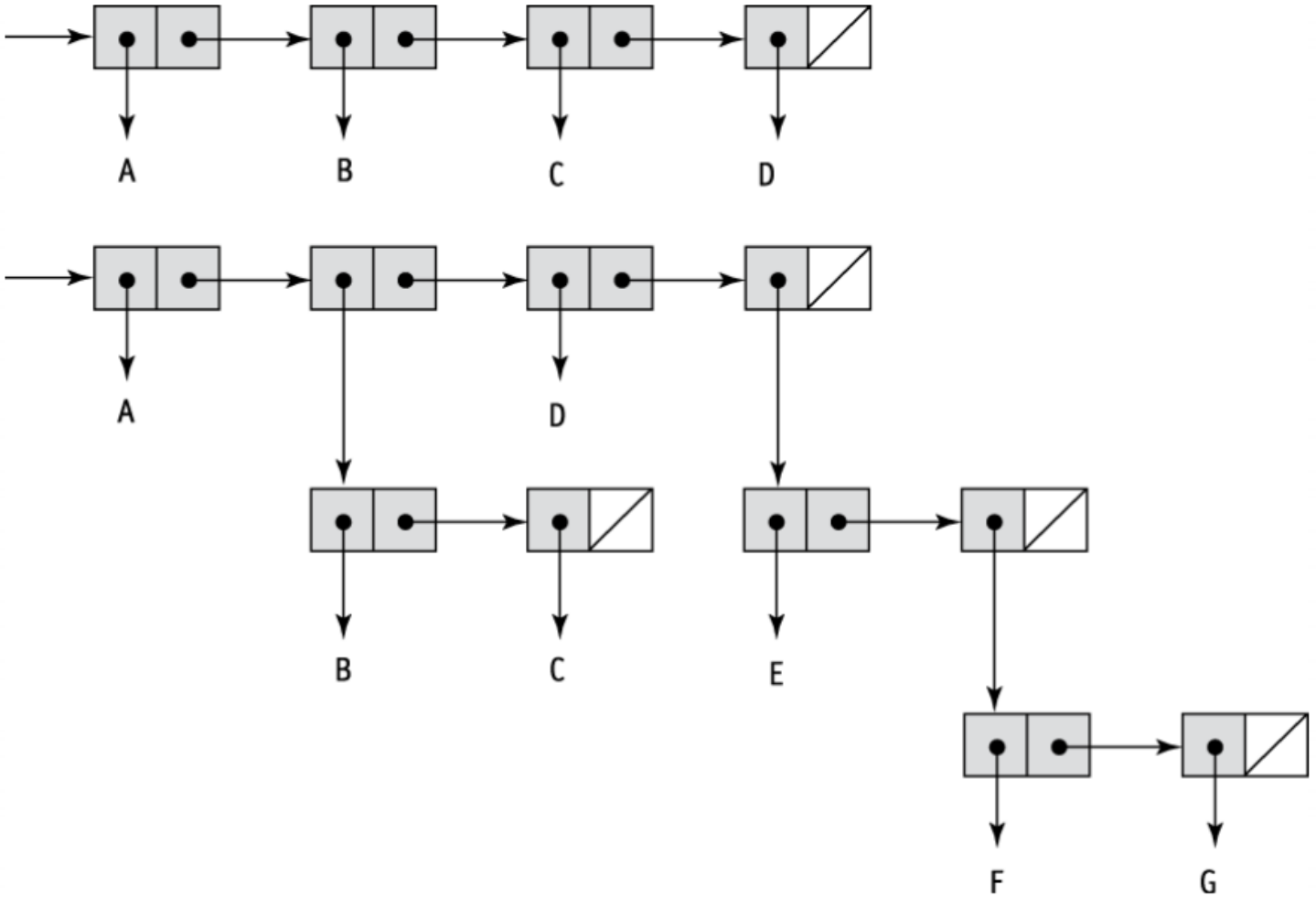
- A primeira linguagem funcional
- Tipos de objetos de dados: originalmente, apenas átomos e listas
- Lista: coleção de sublistas e/ou átomos, entre parênteses

  – por exemplo, (A (B C) D (E (F G)))
- Originalmente, LISP era desprovida de tipos
- Internamente, as listas são armazenadas como estruturas de lista encadeadas, nas quais cada nó possui dois ponteiros e representa um elemento

REPRESENTAÇÃO INTERNA DE DUAS LISTAS EM LISP

- Representando as listas **(A B C D)**

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



INTERPRETAÇÃO EM LISP

UNIVERSIDADE FEDERAL DO CEARÁ

- A notação lambda é usada para especificar funções e definições de funções
- – Por exemplo, se a lista (A B C) é interpretada como dados, é uma simples lista de três átomos, A, B e C
- – Se é interpretada como uma aplicação de função, significa que a função chamada A é aplicada aos dois parâmetros, B e C
- Por exemplo, Se + é uma função que recebe dois parâmetros, então

(+ 5 7)

é avaliada como 12