


Visão geral dos paradigmas declarativos e descritivos

Paradigmas de Programação – BCC/UFRPE
Lucas Albertins – lucas.albertins@deinfo.ufrpe.br

Agenda

- + Contexto
- + Paradigmas Declarativos
 - + Funcional
 - + Lógico
- + Paradigmas Descriptivos
 - + Imperativo – Procedural
 - + Imperativo – Orientado a objeto
 - + Orientado a aspectos

Linguagens de Programação

- + Linguagens de Programação – Objetivo principal:
 - + Abstração de código de máquina
 - + Linguagem de máquina -> Linguagem natural -> componentes de software
 - + Devem ser no mínimo
 - + Universais
 - + Implementáveis
- 
- A word cloud containing various programming language names such as C++, Java, JavaScript, Python, PHP, Perl, Ruby, Swift, Kotlin, Rust, Go, R, Julia, Haskell, OCaml, F#, Scala, Clojure, Erlang, Elixir, Lua, Nim, Zig, Fortran, COBOL, PL/I, ALGOL, Pascal, Basic, Visual Basic, VB.NET, C#, Objective-C, Swift, Kotlin, Rust, Go, R, Julia, Haskell, OCaml, F#, Scala, Clojure, Erlang, Elixir, Lua, Nim, Zig, Fortran, COBOL, PL/I, ALGOL, Pascal, Basic, Visual Basic, VB.NET, C#. The words are arranged in a circular pattern, with some larger than others, set against a dark background.



Paradigmas de Programação

- + Paradigma no dicionário
 - + *s.m. Modelo; exemplo utilizado como padrão a ser seguido; norma.*
 - + *Linguística. Conjunto de termos que podem ser substituídos, entre si, na mesma posição da estrutura da qual fazem parte.*
 - + *Gramática. Tipo de conjugação ou declinação gramatical cujas formas vocabulares podem ser usadas como padrão ou modelo: o verbo amar segue o paradigma da primeira conjugação porque termina em "ar".*
 - + *(Etm. do grego: paradeigma)*

Paradigmas de Programação

- + Estilos de programação
- + Cada paradigma possui um conjunto de conceitos chave
- + Duas grandes categorias de paradigmas:

Declarativos	Descriptivos (Imperativos)
Funcional	Procedural
Lógico	Orientado a Objeto
	Orientado a aspectos

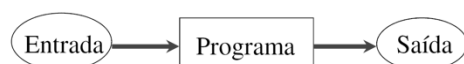
Principais Diferenças

- + Programação Declarativa
 - + Foco no O QUE é executado
 - + Não existe o conceito tradicional de:
 - + Estado
 - + Variáveis
 - + Atribuições
 - + Contexto
 - + Efeitos colaterais
 - + Ordem de execução
- + Programação Descritiva
 - + Foco no COMO é executado
 - + Baseado em uma ordem de execução

Paradigmas Declarativos

O Paradigma Funcional

- + Programas são funções que descrevem uma relação explícita e precisa entre E/S
- + Um mapeamento



Paradigma Funcional

- + Estilo declarativo: não existem os conceitos de: estados, atribuições, contexto, efeitos colaterais e ordem de execução
- + Transparência Referencial
- + Baseado em:
 - + Expressões:
 - + Funções
 - + Polimorfismo parametrizado
 - + Abstração de dados (em algumas linguagens)
 - + Lazy-evaluation

Paradigma Funcional

- + Expressões
 - + Computar novos valores a partir de antigos
 - + Aritméticas, Lógicas, Condicionais
- + Funções
 - + Abstração de expressões
 - + Consideradas valores de primeira classe
 - + Passada como argumento, resultado de outras funções, usada para compor valores, etc
 - + Funções de alta ordem
 - + Recebem função como argumento ou retorna função como resultado (f o g)
- + Polimorfismo parametrizado
 - + Permite operar sobre conjunto de tipos

Paradigma Funcional

- + Abstração de Dados
 - + Apenas nas linguagens mais modernas
 - + Contém **constantes e funções**
- + Lazy Evaluation
 - + Avalia uma expressão apenas quando é usado
 - + Se não for usada, nunca é avaliada
 - + Similar ao curto circuito (& e &&, | e ||)

Exemplo em LISP

```

1: defun factorial (n)
2: (if (<= n 1)
3:     1
4:     (* n (factorial (- n 1))))
  
```

Visão Crítica do Paradigma Funcional

- + Vantagens
 - + Manipulação de programas mais simples:
 - + Prova de propriedades
 - + Transformação (exemplo: otimização)
 - + Transparência referencial (mesma entrada, mesma saída)
- + Desvantagens
 - + "O mundo não é funcional!"
 - + Implementações ineficientes
 - + Mecanismos primitivos de E/S e formatação

Principais Linguagens Funcionais

- + LISP – Uma das primeiras linguagens funcionais
- + ML – Linguagem funcional impura
- + Haskell – linguagem funcional pura
- + Scala – Funcional Orientada a Objetos

Paradigma Lógico

- + Programas são relações entre dados



Paradigma Lógico

- + Estilo declarativo, como no paradigma funcional
- + Baseada na lógica simbólica: proposições
 - + Declarações V ou F sobre objetos
 - + Relações entre os objetos
- + Computação:
 - + Lista de fatos (dados)
 - + Relações entre os dados como hipóteses
 - + Metas a serem inferidas: resolução
 - + Perguntas

Paradigma Lógico

- + Proposições representadas através de **Cláusulas de Horn**

$$u \leftarrow (p \wedge q \wedge \dots \wedge t) \quad \text{OU} \quad (p \wedge q \wedge \dots \wedge t) \rightarrow u$$
- + *u é verdadeiro se p e q e ... e t são verdadeiros*
- + Cláusulas de Horn sem cabeça (lado esquerdo): **Fatos**
 - + pai(joao, maria)
- + Cláusula de Horn com cabeça: **Relações**
 - + avo(joao, maria) <- pai(joao, pedro) ^ pai(pedro, maria)
 - + avo(x,z) <- pai(x,y) ^ pai(y,z)

Paradigma Lógico - Exemplo

- + **Quais alunos provavelmente terão dificuldades nessa disciplina?**
- + **Meta:** Quais alunos provavelmente terão dificuldade na disciplina X?
- + **Relações:** se aluno foi aprovado com menos de 6 nas disciplinas pré-requisitos de X, provavelmente terá dificuldades na disciplina X
- + **Fatos:**
 - + Disciplinas A, B e C são pré-requisitos da disciplina X
 - + Média dos alunos nas disciplinas A, B, C

Paradigma Lógico – Exemplo Prolog

```

1: orbits(mercury, sun). {facts}
2: orbits(venus, sun).
3: orbits(earth, sun).
4: orbits(mars, sun).
5:
6: orbits(moon, earth).
7:
8: orbits(phobos, mars).
9: orbits(deimos, mars).
10:
11: planet(P) <= orbits(P, sun). {rules}
12: satellite(S) <= orbits(S, P), planet(P).
13:
14: ? satellite(S). {query}

```

--- running ---

- satellite(moon)
 - true
- satellite(phobos)
 - true
- satellite(deimos)
 - true
- satellite(mercury)
 - false

Visão Crítica do Paradigma Lógico

- + Vantagens
 - + Em princípio, todas do paradigma funcional
 - + Permite concepção da aplicação em um alto nível de abstração (através de associações entre E/S)
- + Desvantagens
 - + Em princípio, todos do paradigma funcional
 - + Linguagens usualmente não possuem tipos

Aplicações do Paradigma Lógico

- + Bancos de dados
 - + Armazenam dados: fatos
 - + Declarativo: preocupação com os dados trazidos e não em como são trazidos
 - + Equivalência direta:

```

altoRisco(A) <- homem(A), fatorDeRisco(A, sedentarismo),
               fatorDeRisco(A, obesidade).

```

Equivalente a

```

select * from pacientes where sexo = 'Masculino' and sedentarismo =
true and obesidade = true

```

Aplicações do Paradigma Lógico

- + Sistemas especialistas
 - + Tirar conclusões a partir de fatos e regras
 - + Capacidade de *trace*: Justificativa
- + Educação
 - + Prova de teoremas
 - + Ensino e prática de raciocínio lógico
 - + Ensino matemático
 - + Ensino diverso
 - + Utilizando *trace*

Paradigmas declarativos

Exercício: Quais outros exemplos de aplicação prática podemos utilizar linguagens funcionais e lógicas?

Paradigmas Descritivos

Paradigma Imperativo

- + Programas centrados no conceito de um estado (modelado por variáveis) e ações (comandos) que manipulam o estado



Paradigma Imperativo

- + Objetivo: abstração
 - + Endereços de memória -> variáveis e tipos
 - + Conjunto de dados -> tipos compostos, arrays e registros
 - + Conjunto de comandos -> subrotinas, funções e procedures
- + Também denominado procedural
- + Primeiro paradigma a surgir e ainda é o dominante

Exemplo de Programa - C

```

1: #include <stdio.h>
2:
3: int main() {
4:     int n, /* guarda o numero dado */
5:     i, /* auxiliar */
6:     nfat; /* valor calculado */
7:
8:     printf ("\nDigite um inteiro nao-negativo: ");
9:     scanf ("%d", &n);
10:    nfat = 1;
11:    for (i = n; i > 1; i--) {
12:        nfat = nfat * i;
13:    }
14:    printf ("O valor de %d!= %d\n", n, nfat);
15:    return 0;
16: }
```

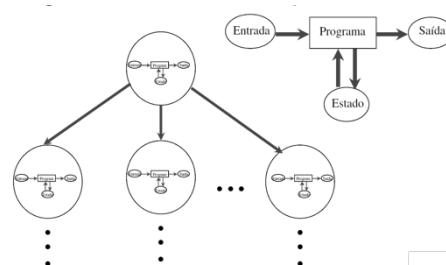
Visão Crítica do Paradigma Imperativo

- + Vantagens
 - + Eficiência (embute modelo de Von Neumann)
 - + Modelagem "natural" de aplicações do mundo real
 - + Paradigma dominante e bem estabelecido
- + Desvantagens
 - + Relacionamento indireto entre E/S resulta em:
 - + Dificil legibilidade
 - + Erros introduzidos durante manutenção
 - + Descrições demasiadamente operacionais focalizam o **como** e não o **que**

Paradigma Orientado a Objeto

- + Subclassificação do imperativo
- + A diferença basicamente de metodologia quanto à concepção e modelagem do sistema
- + Mais um nível de abstração:
 - + Aplicação estruturada em módulos (classes) que agrupam um (ou um conjunto de) estado e operações (métodos) sobre este
- + Classes podem ser estendidas e/ou usadas como tipos (cujos elementos são objetos)

Modelo Computacional do Paradigma Orientado a Objetos



Exemplo OO

```

1: public class Number {
2:     int num;
3:     Number (int i) {
4:         num = i;
5:     }
6:     public int getFatorial() {
7:         ...
8:         return nfat;
9:     }
10: }
11:
12:
13: public static void main(String[] args) {
14:     int number = Integer.parseInt(args[0]);
15:     Number fat = new Number(number);
16:     int result = fat.getFatorial();
17:     System.out.println(result);
18: }
19:
20: }

```

Visão Crítica do Paradigma OO

- + Vantagens
 - + Todas as do estilo imperativo
 - + Classes estimulam projeto centrado em dados: modularidade, reuso e extensibilidade
 - + Aceitação comercial crescente
- + Desvantagens
 - + Semelhantes aos do paradigma imperativo, mas amenizadas pelas facilidades de estruturação

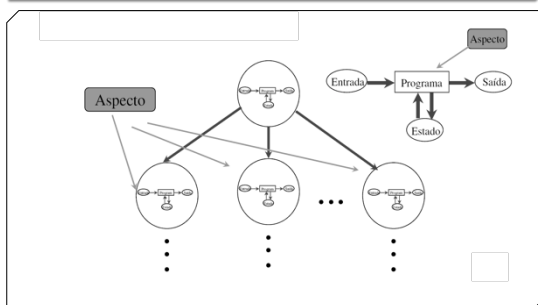
Paradigma Orientado a Aspecto

- + Não é um paradigma no sentido estrito
- + A diferença é mais de metodologia quanto à concepção e modelagem do sistema
- + Nova forma de modularização:
 - + Para "requisitos" que afetam várias partes de uma aplicação

Paradigma Orientado a Aspecto

- + Aplicações estruturadas em **módulos (aspectos)** que agrupam **pontos de interceptação de código (pointcuts)** que **afetam outros módulos (classes)** ou outros aspectos, definindo **novo comportamento (advice)**
- + Aspectos podem ser estendidos e/ou usados como tipos

Modelo Computacional do Paradigma Orientado a Aspectos



Exemplo AspectJ

```

1: public aspect Imprime {
2:
3:     public pointcut fatorialCalls():
4:         call(int Number.getFatorial());
5:
6:     before(): fatorialCalls() {
7:         System.out.println("Calculando
8:         Fatorial...");
9:     }
10: }

```

