UFRPE
Universidade
Federal Rural
de Pernambuco

Paradigmas de Programação

Aula 05
Visão Geral de Paradigmas Descritivos

Kellyton Brito
kellyton.brito@gmail.com
20/10/2017

Contexto e Motivação

- Paradigmas de Progamação
  - Modelagem Computacional do Mundo Real

- Paradigma Funcional
- Paradigma Lógico
- Paradigma Imperativo
  - Orientado a Objetos
    - Orientado a Aspectos

Paradigmas e Linguagens de Programação

- Objetivo principal:
  - Abstração de código de máquina
  - Linguagem de máquina -> Linguagem natural -> componentes de software

- 2 Grandes Grupos

Declarativas	Descritivas
Funcionais	Imperativas
Lógicas	Orientado a Objetos
	Orientado a Aspectos

Diferença entre os grupos de paradigmas

- **Programação descritiva**
  - Foco no COMO é executado
  - Baseado em uma ordem de execução
  - Baseado em:
    - Estados,
    - Variáveis,
    - Atribuições,
    - Contexto,
    - Efeitos colaterais e
    - Ordem de execução
- **Programação declarativa**
  - Foco no QUÊ é executado
  - Programa lido e executado todo de uma vez
  - Não existe conceitos tradicionais de: Estados, Variáveis, Atribuições, Contexto, Efeitos colaterais e Ordem de execução

UFRPE
Universidade Federal Rural de Pernambuco

Paradigmas Descritivos

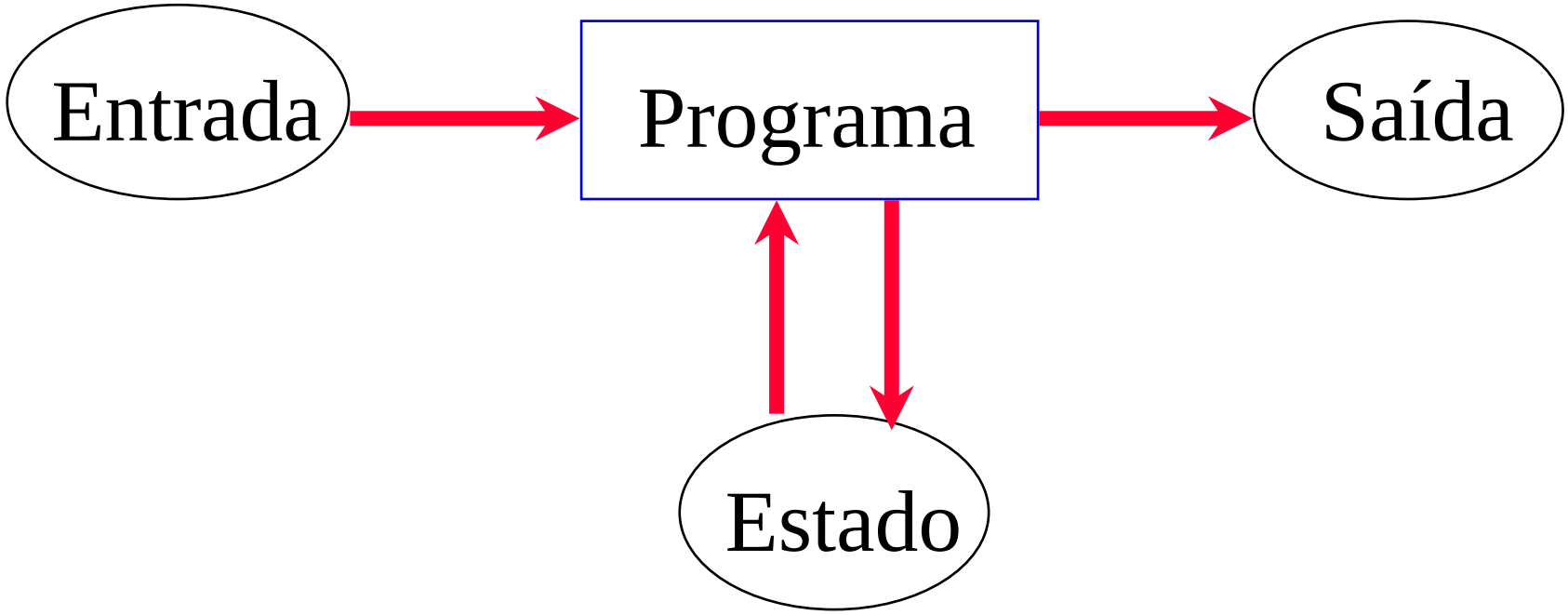
UFRPE
Universidade
Federal Rural
de Pernambuco

Paradigmas Descritivos

Paradigma Imperativo

O 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

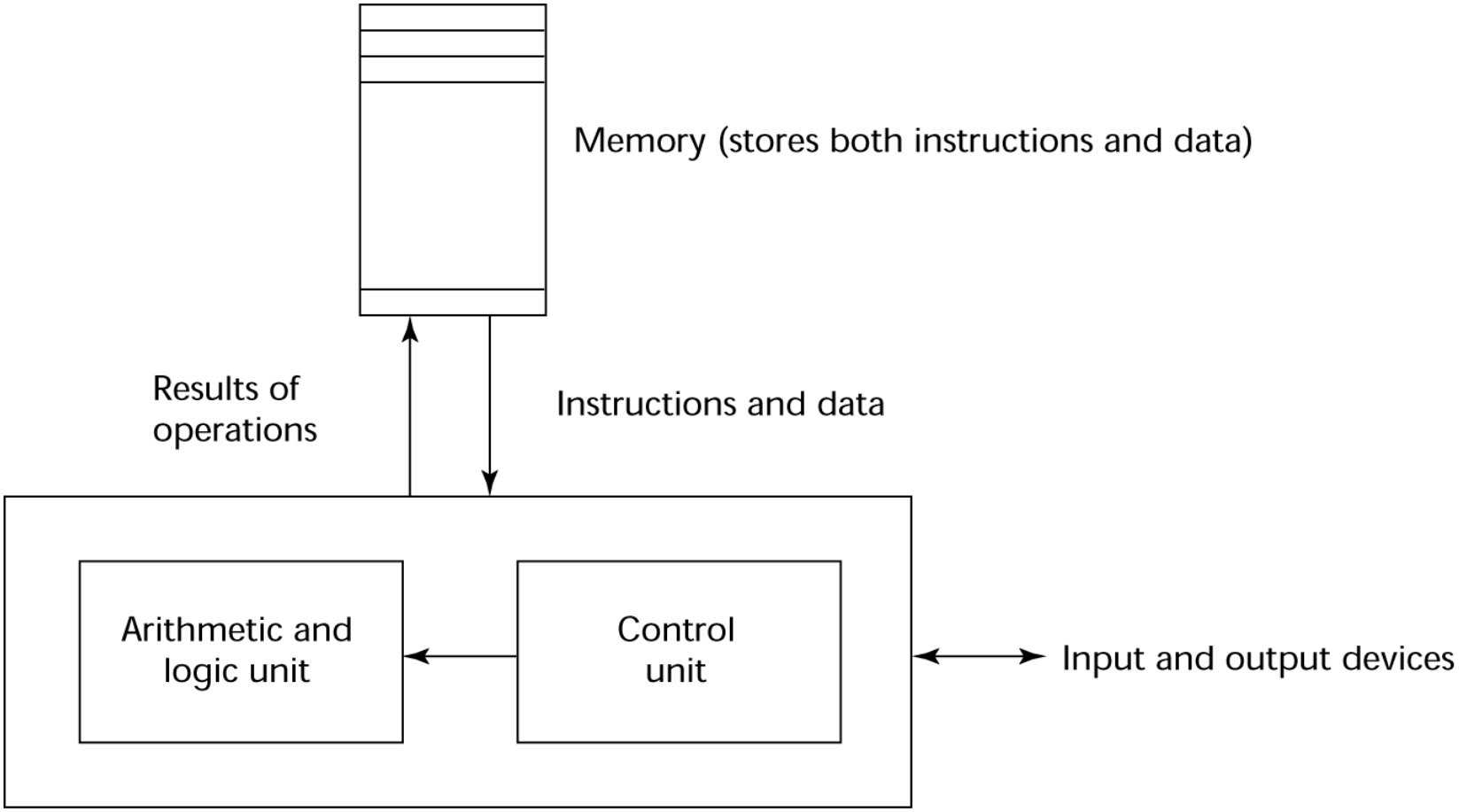
Exemplo de Programa - C

```
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

Arquitetura de Von Neumann



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:
    - Difícil legibilidade
    - Erros introduzidos durante manutenção
    - Descrições demasiadamente operacionais focalizam o **como** e não **o que**

UFRPE
Universidade
Federal Rural
de Pernambuco

Paradigmas Descritivos

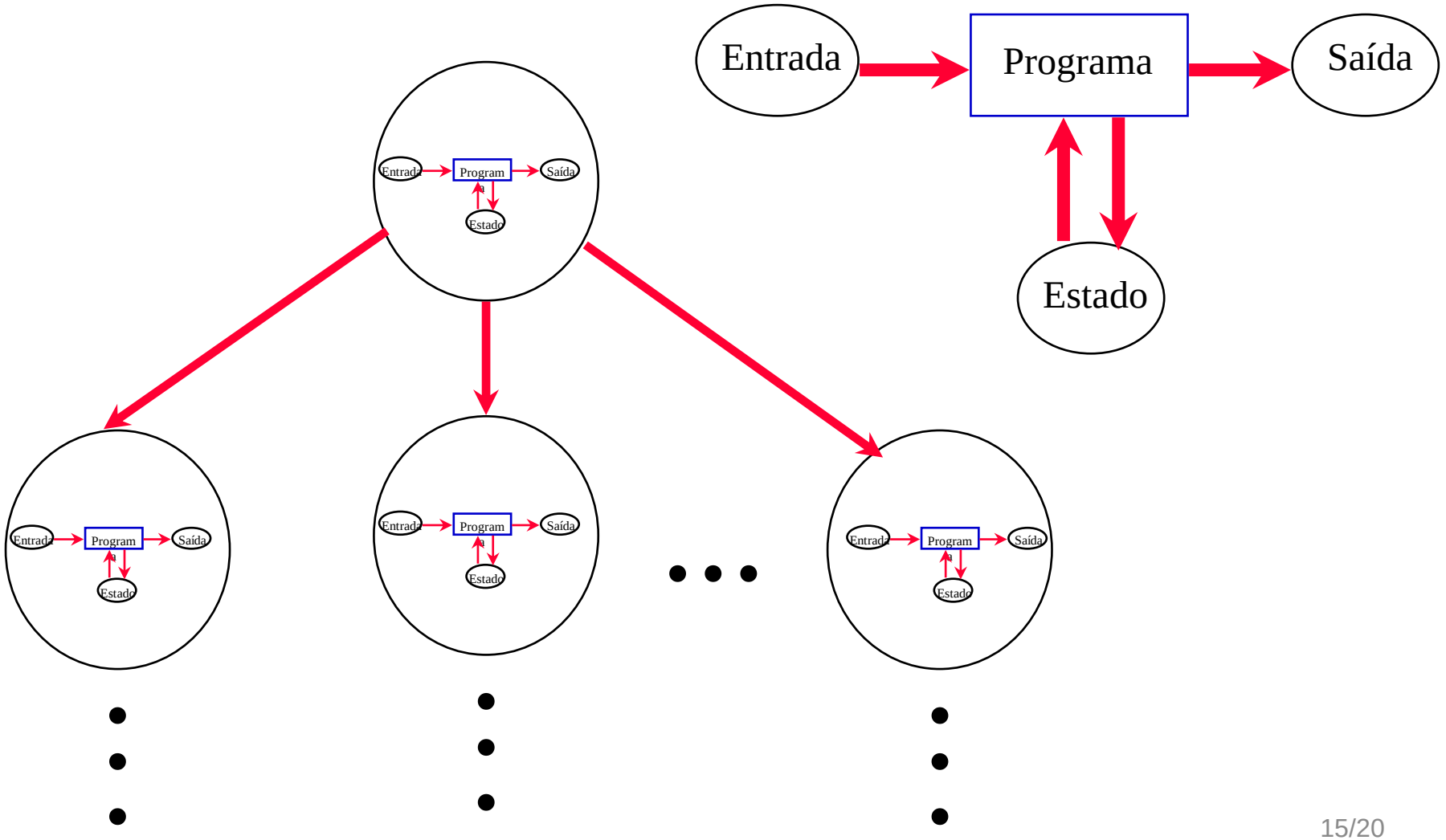
Paradigma Imperativo

Orientado a Objetos

O Paradigma Orientado a Objetos

- Não é um paradigma no sentido estrito: 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 de Programa OO - Java

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

UFRPE
Universidade Federal Rural de Pernambuco

Paradigmas Descritivos

Paradigma Imperativo

Orientado a Objetos

Orientado a Aspectos

O Paradigma Orientado a Aspectos

- Não é um paradigma no sentido estrito

- A diferença é mais de metodologia quanto à concepção e modelagem do sistema

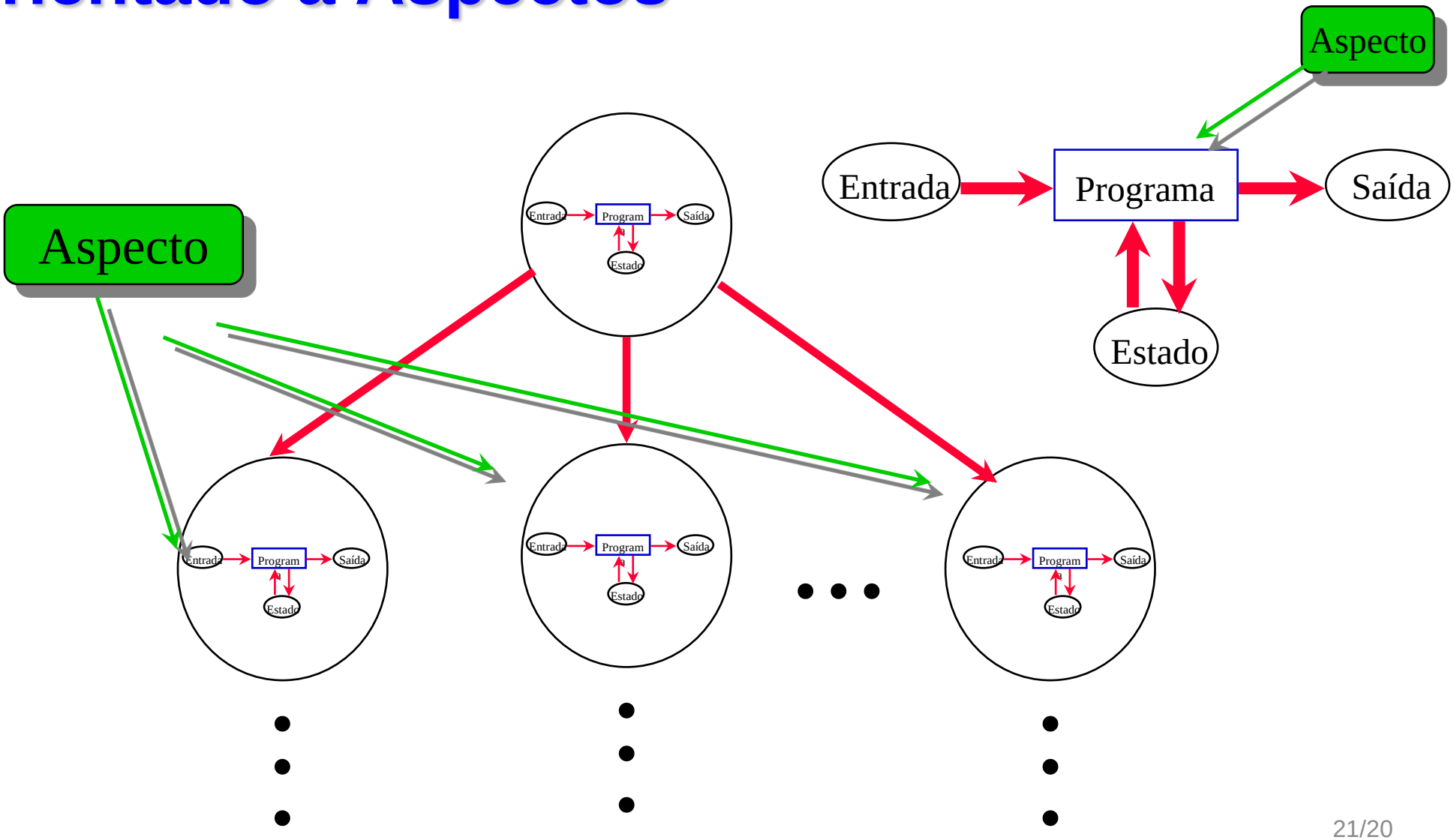
- Nova forma de modularização:
  - Para “requisitos” que afetam várias partes de uma aplicação

O Paradigma Orientado a Aspectos

- 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 de Programa OA - AspectJ

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

Visão Crítica do paradigma OA

- Vantagens
  - Todas as do paradigma OO
  - Útil para modularizar conceitos que a Orientação a Objetos não consegue (*crosscutting concerns*)
    - Em especial, aqueles ligados a requisitos não funcionais
  - Aumenta a extensibilidade e o reuso
- Desvantagens
  - Semelhantes aos do OO
  - Ainda é preciso diminuir a relação entre classes e aspectos
  - Problemas de conflito entre aspectos que afetam a mesma classe

Outros Paradigmas

Outras classificações de paradigmas

- Programação concorrente
  - Execução paralela de 2 ou mais processos (*thread*s)
  - Sincronização intra-processos
  - Acesso sincronizado a dados compartilhados
  - Escrita sincronizada a dados compartilhados

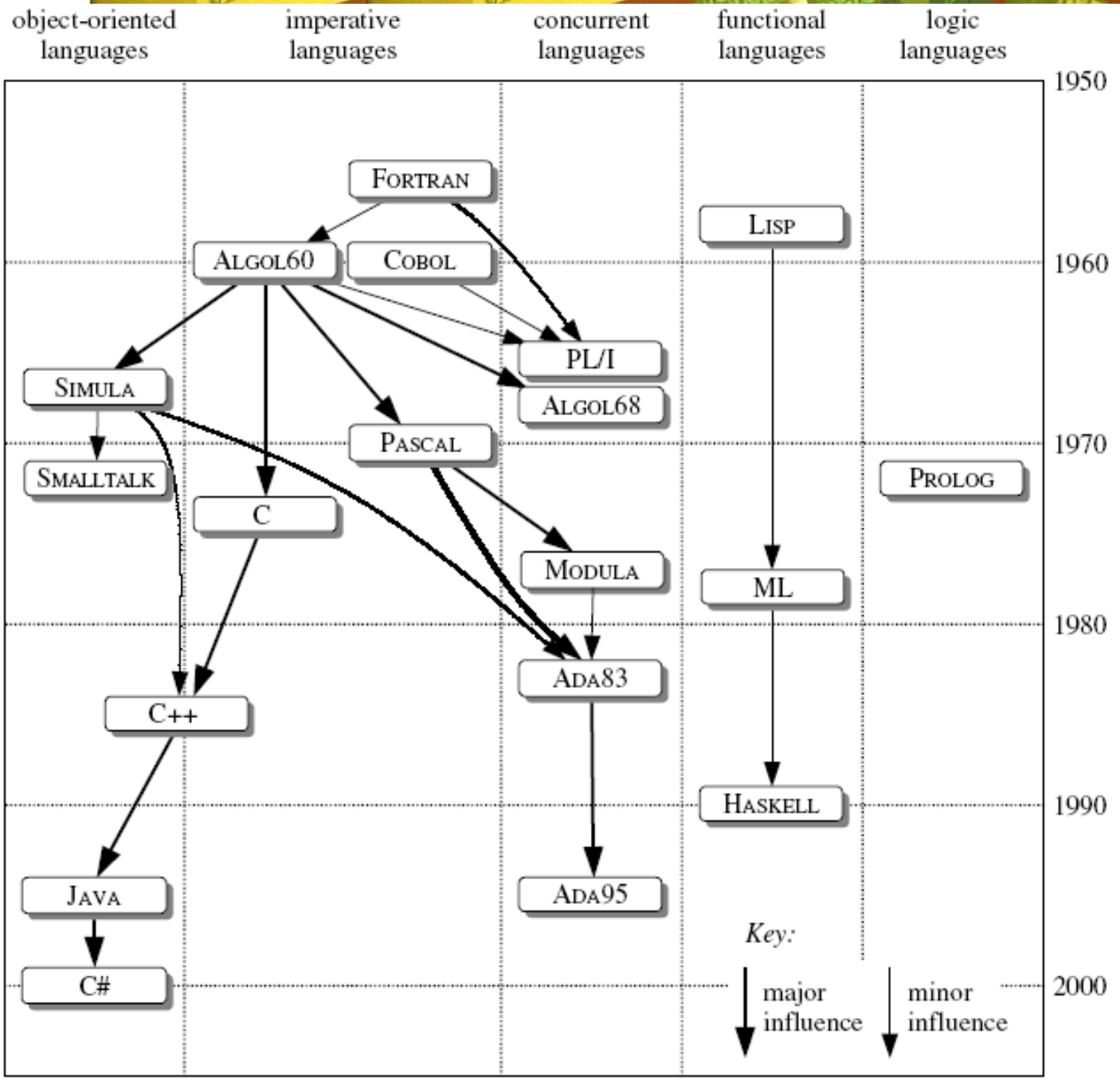
  - Concorrência incluída como funcionalidade adicional das linguagens/paradigmas atuais, como Java, C/C++...

  - Ex.: Desenvolvimento de um SO

Outras classificações de paradigmas

- *Scripting*
  - Utilização de scripts como “código cola”
    - Comunicação/junção/integração entre:
      - Sistemas distintos
      - Partes distintas do mesmo sistema
    - Linguagens/tecnologias/frameworks diferentes
  - Aplicações de alto nível
  - Rápido desenvolvimento e evolução dos scripts
  - Baixo requisito de eficiência

  - Ex.:
    - Script de build/deploy: compila, gera controle de qualidade, gera documentação, empacota e faz o deploy;
    - “Links” entre páginas/requisições web e o acesso à camada de dados



Fonte: David Watt, Programming Language Design Concepts, Ed. John Wiley & Sons

Novo “tipo” de paradigma

facebook

amazon web services™

twitter

Dropbox

- Troca da camada base:
  SO → API de terceiros