Capítulo 6:
Linguagens de Programação

Ciência da Computação: Uma Visão Abrangente
11a Edição

by
J. Glenn Brookshear

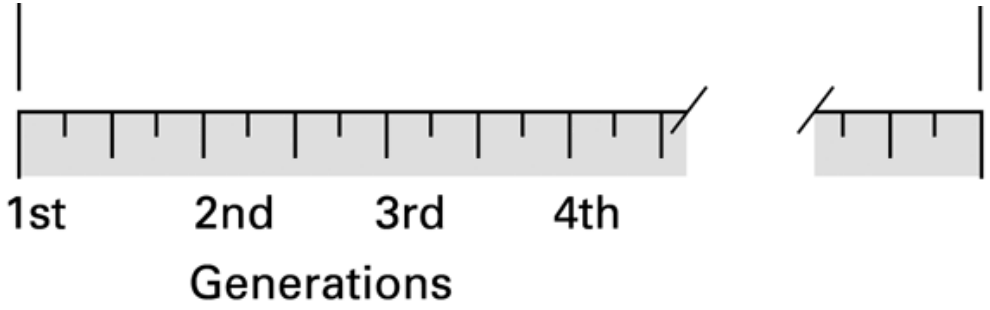
Capítulo 6: Linguagens de programação

- 6.1 Perspectiva histórica
- 6.2 Conceitos de programação tradicionais
- 6.3 Unidades Procedurais
- 6.4 linguagem de Implementação
- 6.5 Programação orientada a objetos
- 6.6 Programação Concorrente

Gerações de linguagens de programação

Problemas solucionados em um ambiente no qual o humano deve estar em conformidade com as características do hardware disponível

Problemas solucionados em um ambiente no qual a máquina deve estar em conformidade com as características do humano



Segunda geração: linguagem Assembly

- Um sistema mnemônico para representar as instruções de máquina
  - Nomes mnemônicos para op-codes
  - Identificadores: Nomes descritivos para locais de memória, escolhidos pelo programador

Características da linguagem assembly

- Correspondência uma-a-uma entre instruções de máquina e instruções de montagem
  - Programador deve pensar como a máquina
- Inerentemente dependente de cada máquina
- Convertido em linguagem de máquina por um programa chamado um Assembler (montador)

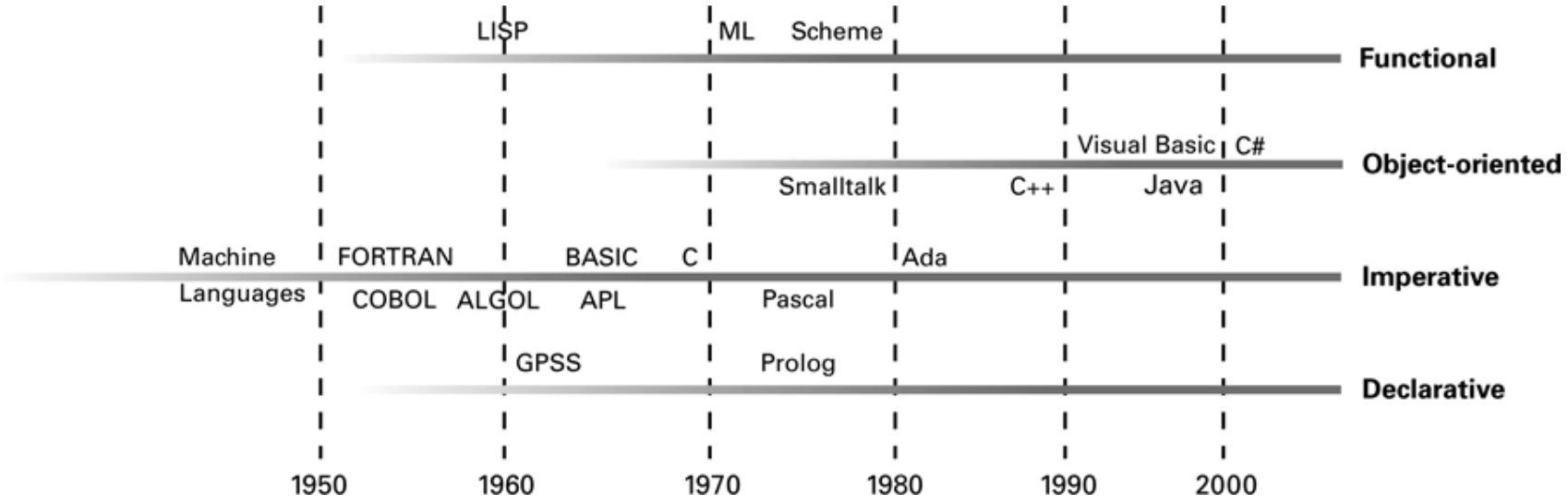
Exemplo de programa

Linguagem de máquina	Linguagem assembly
156C	LD R5, Price
166D	LD R6, ShipCharge
5056	ADDI R0, R5 R6
30CE	ST R0, TotalCost
C000	HLT

Linguagens de Terceira Geração

- Usa primitivas de alto nível
  - Semelhante ao nosso pseudocódigo no capítulo 5
- Predominantemente independente da máquina
- Exemplos: FORTRAN, COBOL
- Cada primitiva corresponde a uma sequência de instruções de linguagem de máquina
- Convertida em linguagem de máquina por um programa chamado um **compilador**

A evolução dos paradigmas de programação



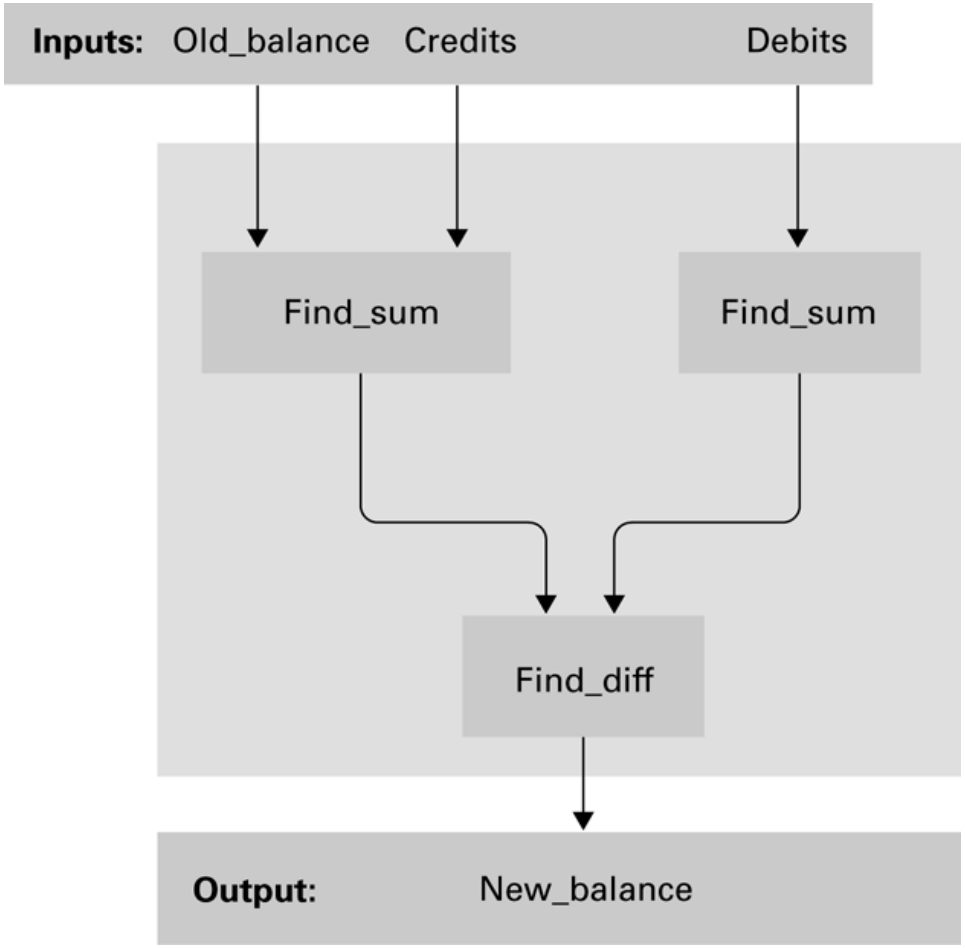
Paradigmas

- **Imperativo** – desenvolvimento de uma sequencia de comandos que, quando executados, manipulam os dados para produzir o resultado esperado
- **Declarativo** – o programador não descreve comandos, e sim o problema a ser resolvido
  - Pela sua complexidade, tipicamente é usado para previsões ou simulações em ambientes específicos

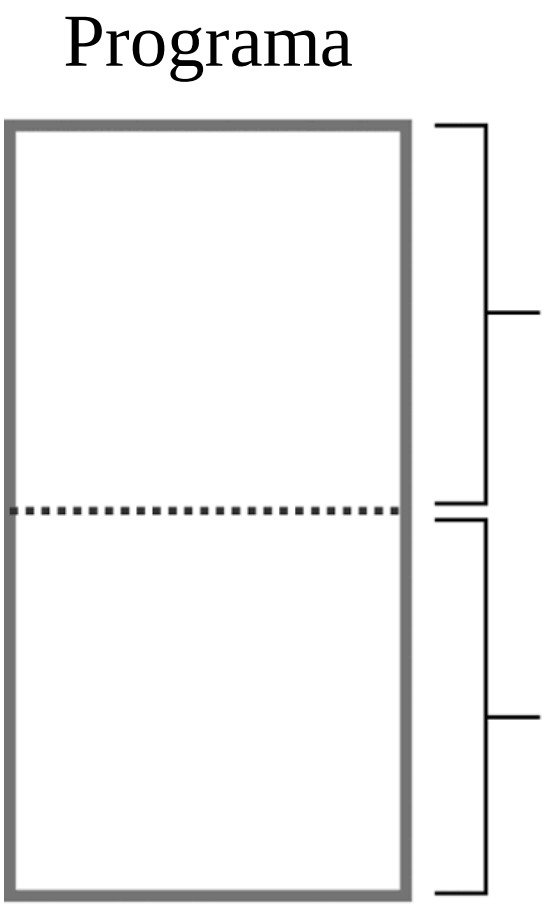
Paradigmas

- **Funcional** – um programa é uma entidade que recebe entradas e produz saídas. Um programa é feito através da conexão de funções
- **Orientado a Objetos** – Um programa é uma coleção de objetos capazes de realizar ações e solicitar ações de outros objetos. Os procedimentos de cada objeto são chamados de métodos e a descrição das propriedades de um objeto de classe. Um objeto criado a partir de uma classe é uma instância

Uma função para calcular saldo através do talão de cheques, construída a partir de funções mais simples



A composição de um programa imperativo



A primeira parte consiste em sentenças de declaração que descrevem os dados manipulados pelo programa

A segunda parte consiste em sentenças imperativas descrevendo a ação a ser realizada

Tipos de dados

- Inteiro: Números inteiros
- Real (float): Números com frações
- Caractere: Símbolos
- Booleano: Verdadeiro/Falso

Declarações de variável

```
float    Length, Width;
int      Price, Total, Tax;
char     Symbol;
```

Uma matriz bidimensional com duas linhas e nove colunas

Pontuações

`Scores (2,4)`
Em FORTRAN, em que os índices começam com 1

`Scores [1] [3]`
Em C e suas derivadas, em que os índices começam com 0

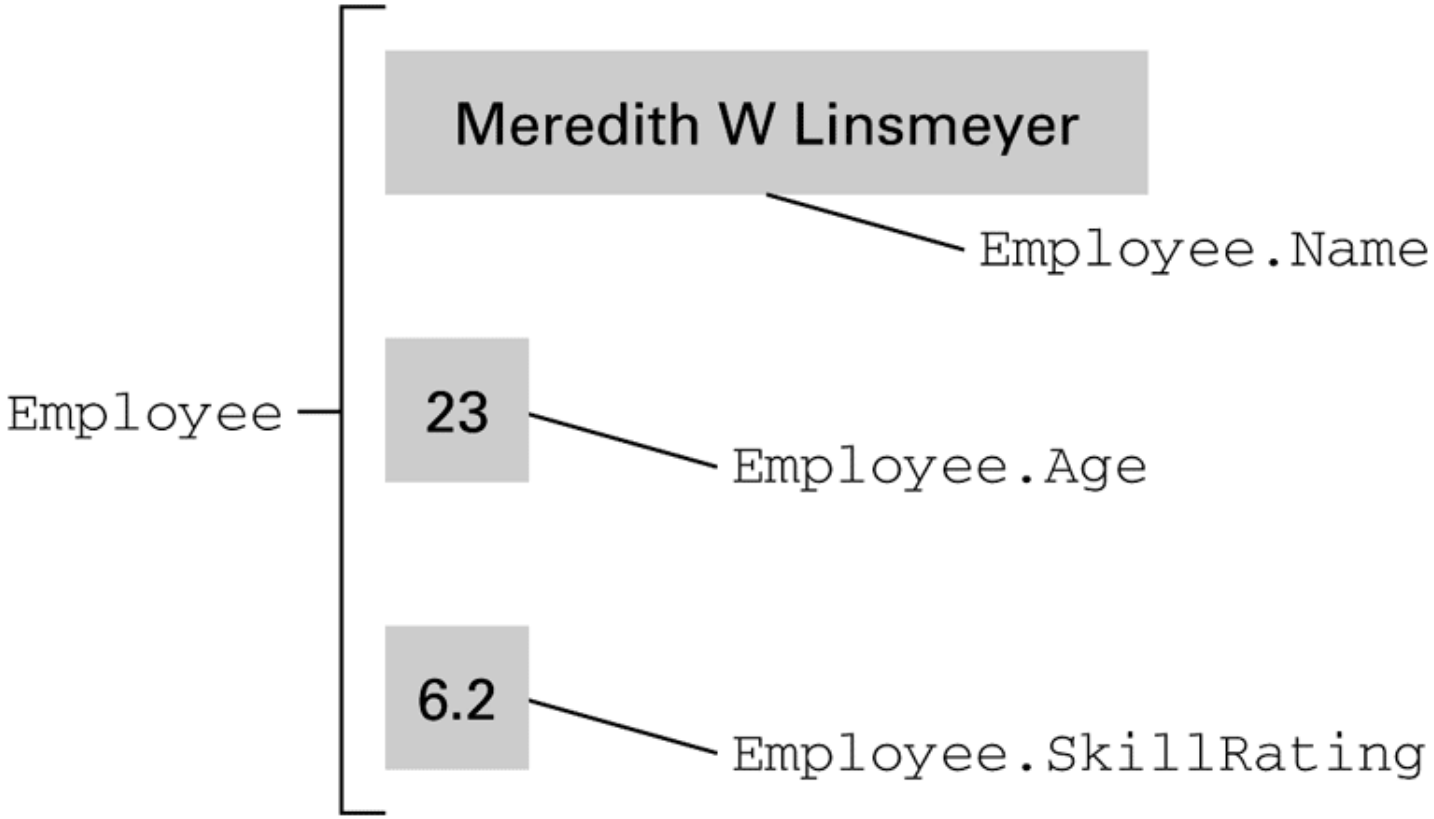
Declaração em FORTRAN:

```
INTEGER Scores (2,9)
```

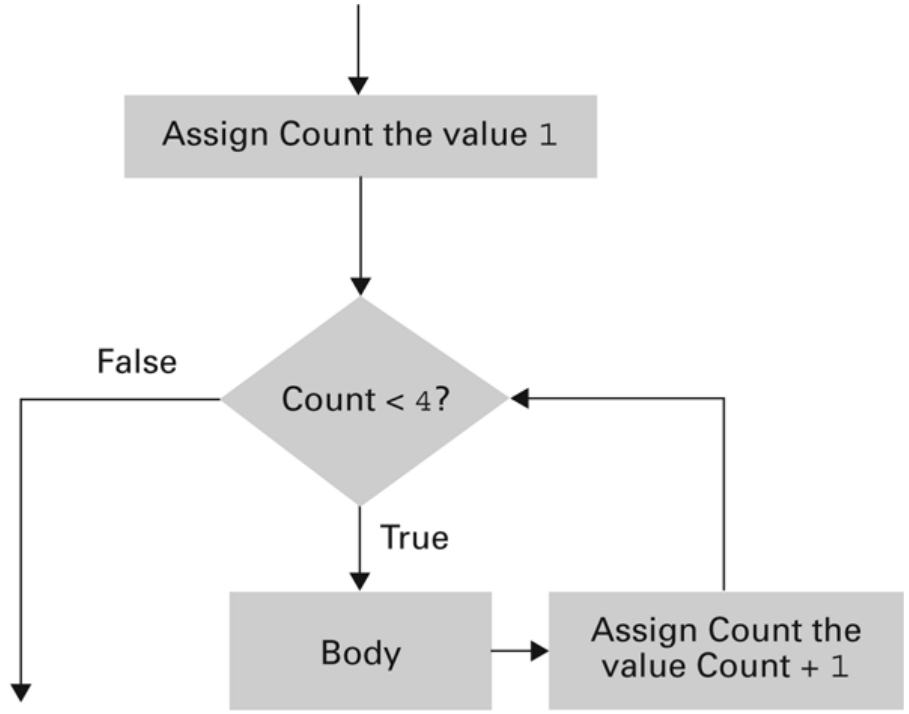
Declaração em C:

```
int Scores [2] [9];
```

A estrutura conceitual do tipo agregado empregado (Employee)



A estrutura do loop FOR (para) e a sua representação em C++, c# e Java

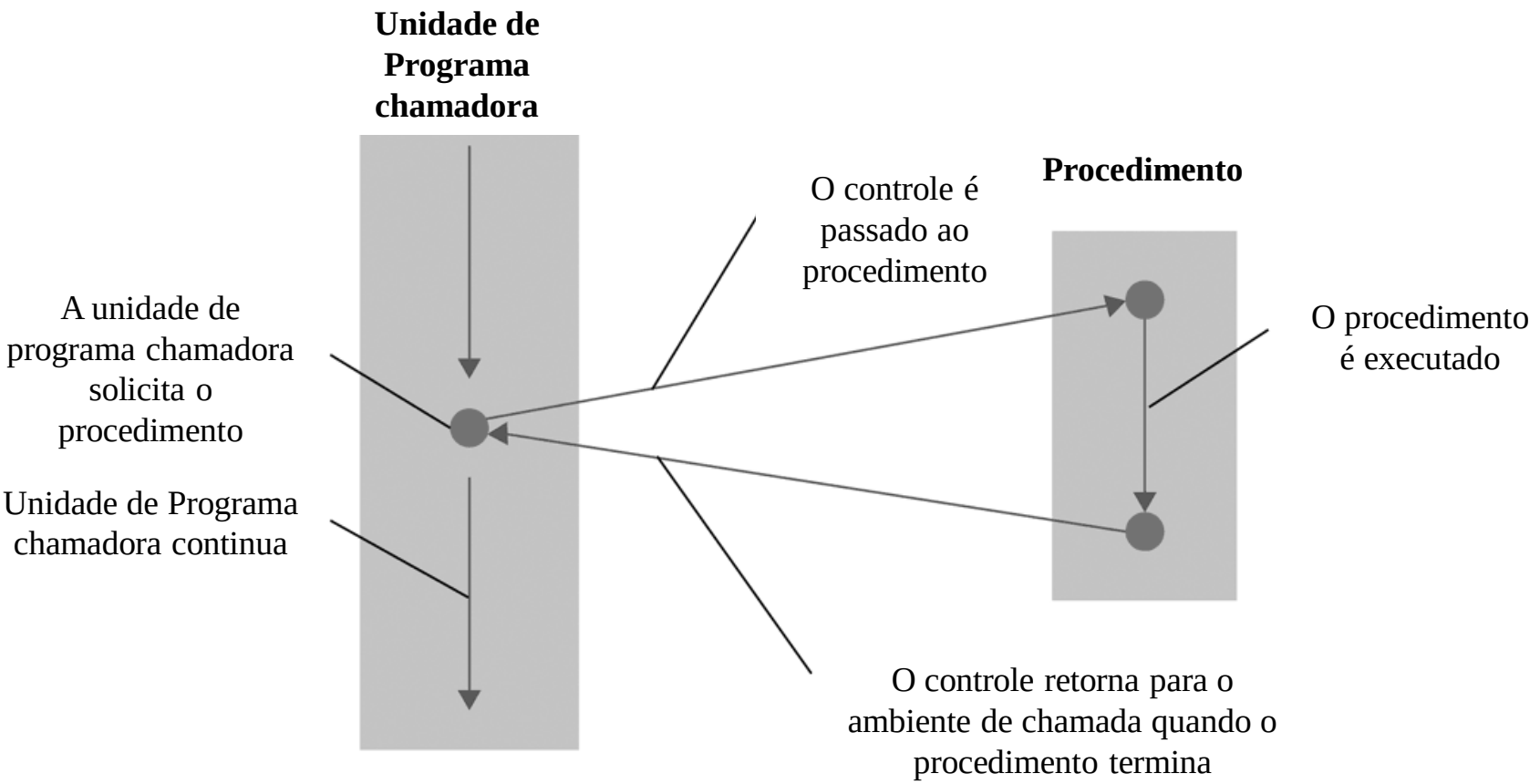



```
for (int Count = 1; Count < 4; Count++)
    body ;
```

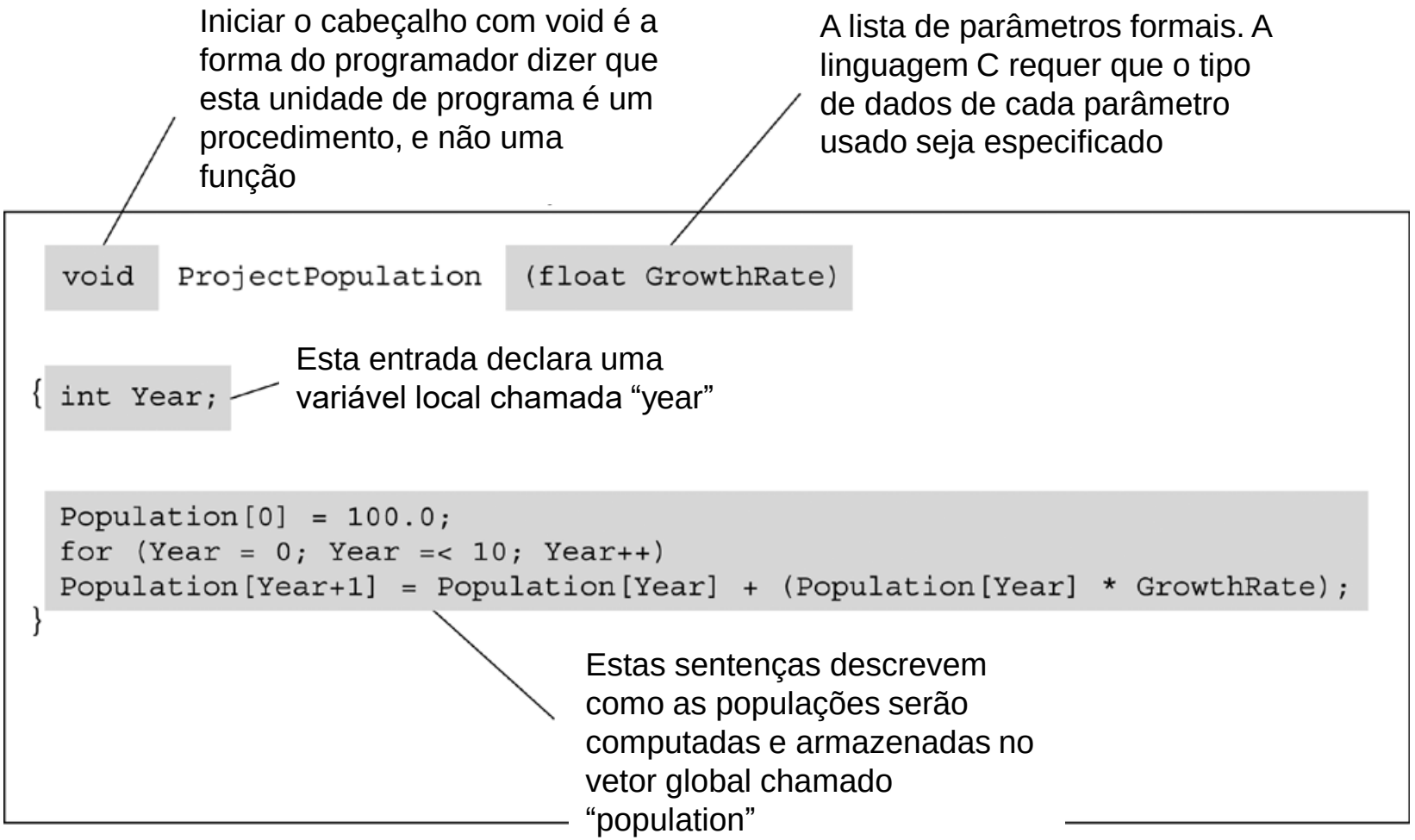
Unidades Procedurais

- Variáveis Locais versus variáveis globais
- Parâmetros Formais versus parâmetros reais
- Passagem de parâmetros por valor versus por referência
- Procedimentos versus Funções

O fluxo de controle que envolve um procedimento

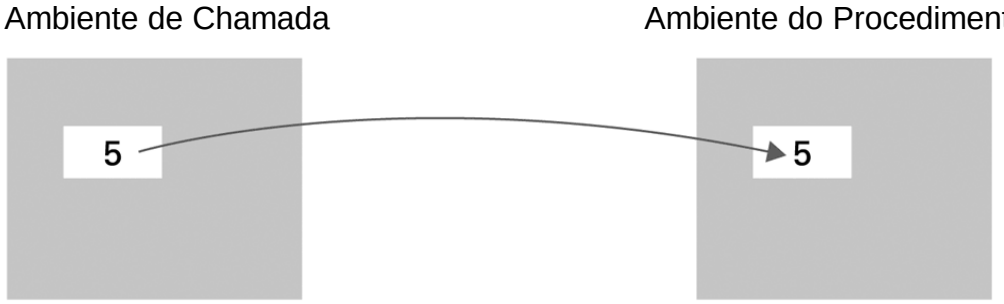


O procedimento ProjectPopulation, escrito em linguagem de programação C



Execução de um procedimento passando parâmetros por valor

a. Quando o procedimento é chamado, uma cópia dos dados é passada ao procedimento...

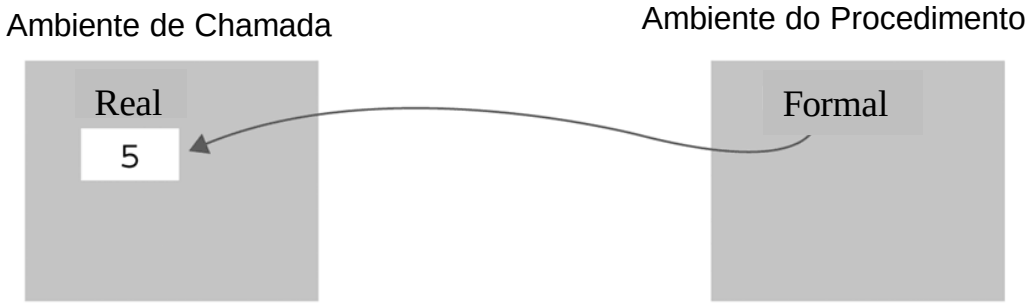


b. E o procedimento manipula esta cópia...

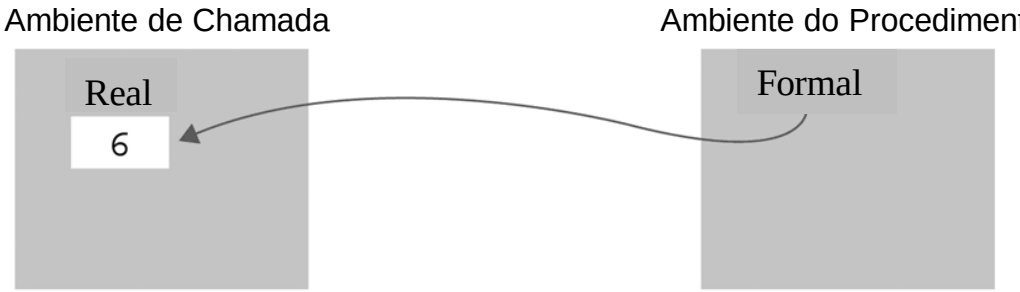
c. Então, quando o procedimento termina, o ambiente de chamada não foi modificado

Execução de um procedimento passando parâmetros por referência

a. Quando o procedimento é chamado, o parâmetro formal torna-se uma referência ao parâmetro real

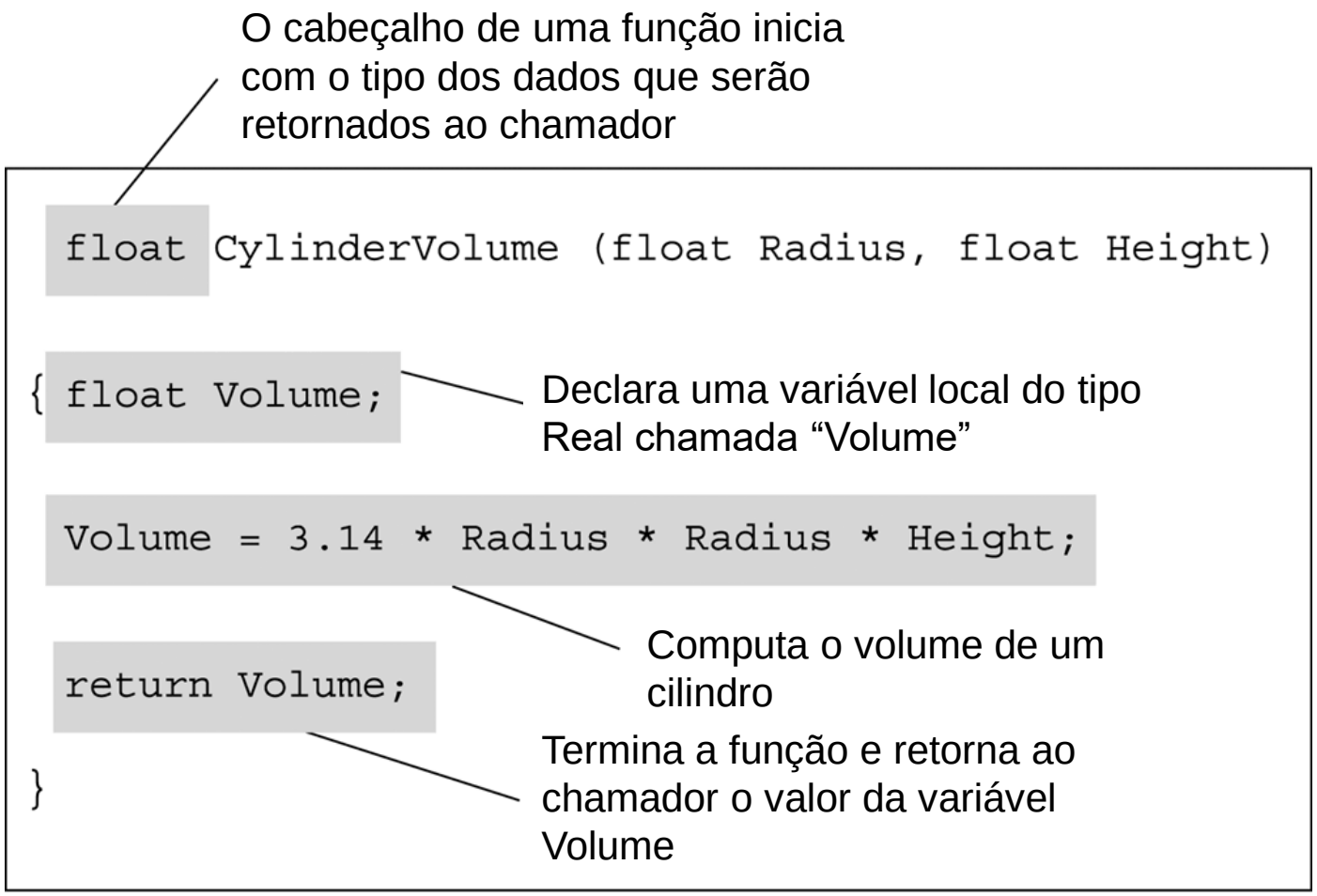


b. Assim, as mudanças solicitadas pelo procedimento são feitas sobre o parâmetro real

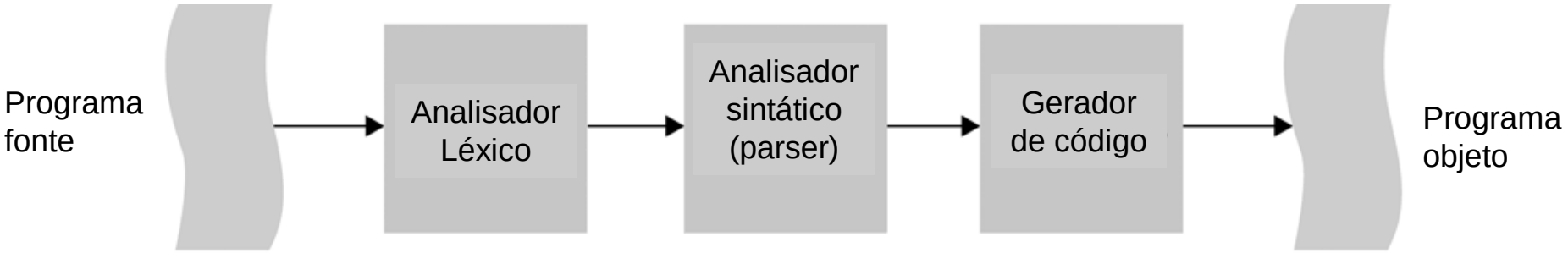


c. Desta forma, as mudanças são preservadas após o procedimento ter se encerrado.

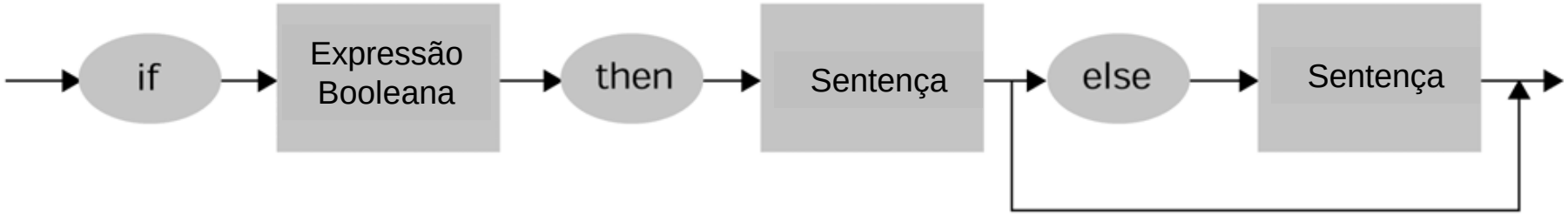
A função CylinderVolume, escrita em linguagem de programação C



O processo de “tradução” de um programa



Um diagrama de sintaxe da instrução if-then-else de pseudocódigo

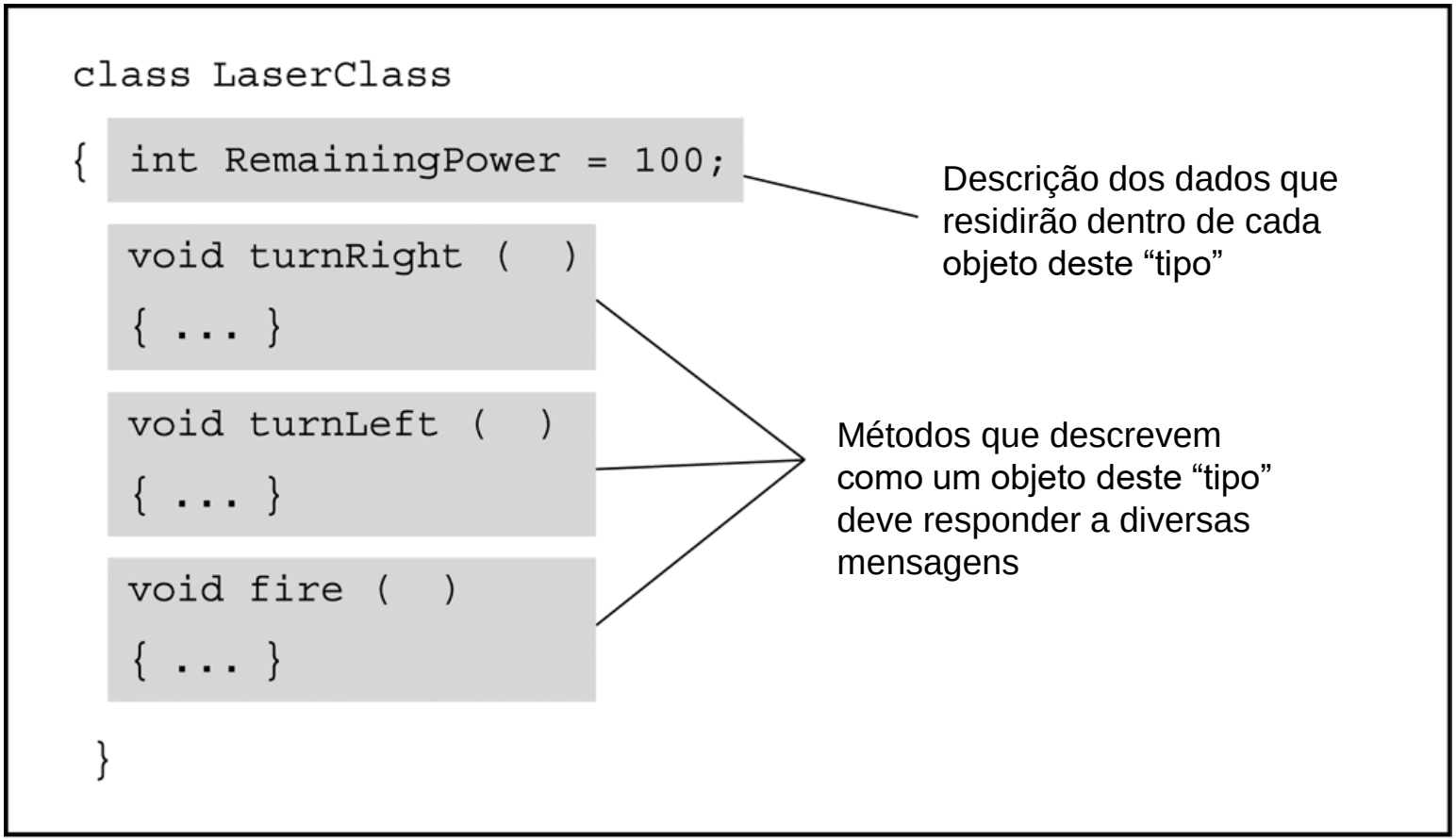


Objetos e Classes

- **Objeto:** Unidade de programa ativo que contém os dados e procedimentos
- **Classe:** Um modelo a partir do qual os objetos são construídos

Um objeto é chamado de uma instância de uma classe.

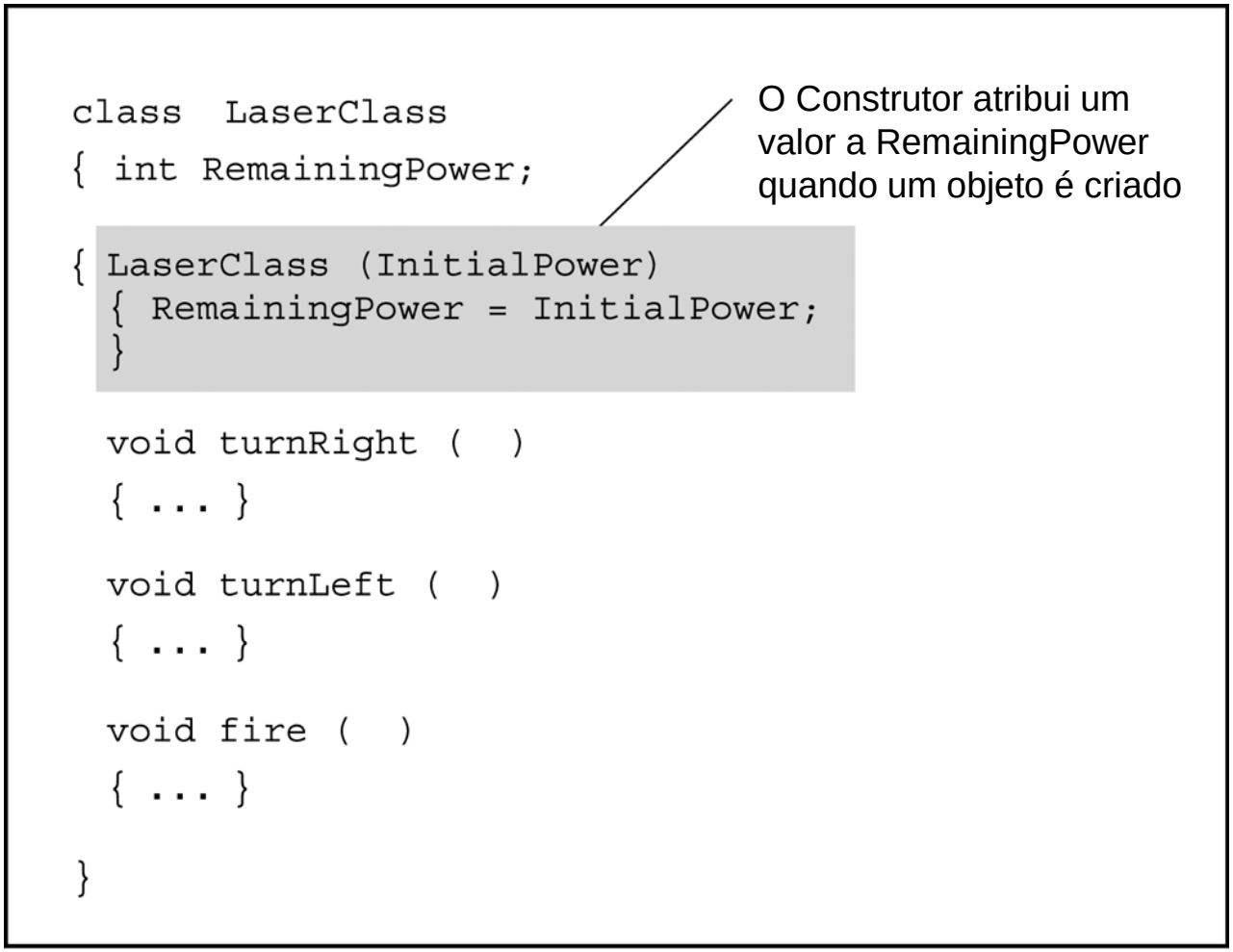
A estrutura de uma classe descrevendo uma arma laser em um jogo de computador



Componentes de um objeto

- **Variável de instância:** Variável dentro de um objeto
  - Mantém informações de dentro do objeto
- **Método:** Procedimento dentro de um objeto
  - Descreve as ações que o objeto pode executar
- **Construtor:** Método especial usado para inicializar um novo objeto quando ele é primeiro construído

Figure 6.21 A class with a constructor

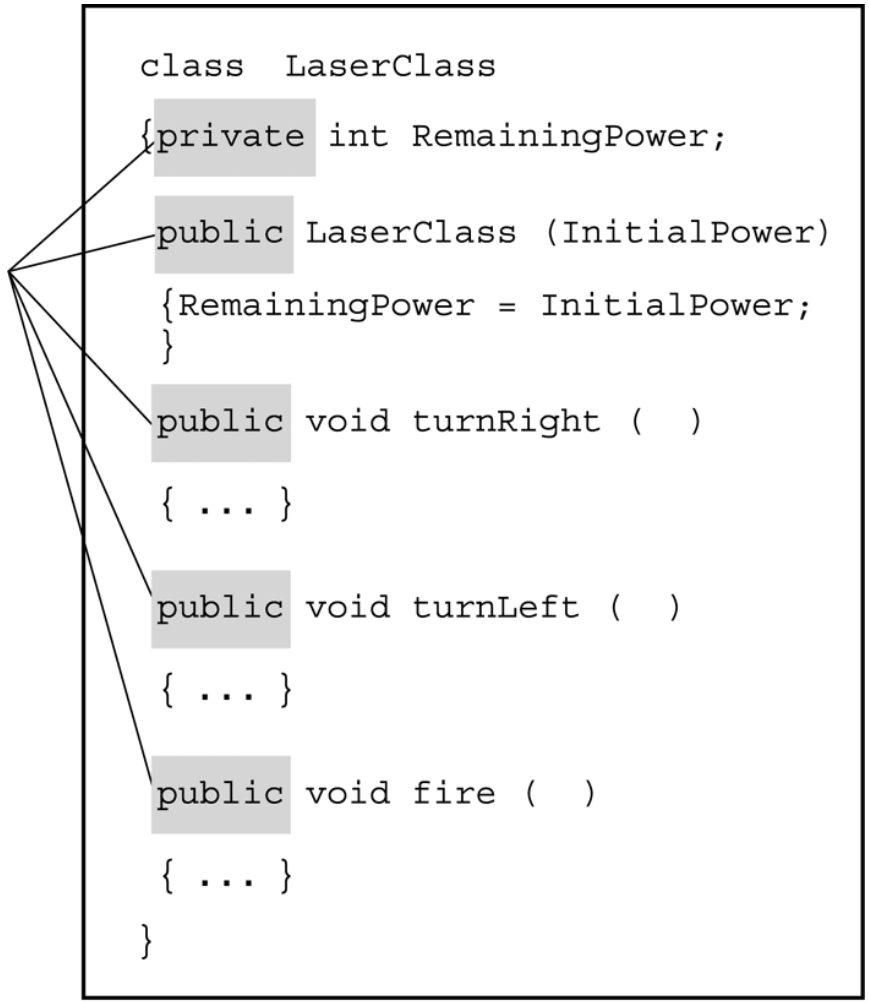


Integridade do objeto

- **Encapsulamento:** Uma maneira de restringir o acesso aos componentes internos de um objeto
  - Private
  - Public

A definição de LaserClass usando encapsulamento e como ela apareceria em um programa Java ou c#

Componentes na classe são designados como públicos ou privados, dependendo de se devem ser acessíveis ou não a partir de outras unidades de programas



```
class  LaserClass
{private int RemainingPower;

public LaserClass (InitialPower)
 {RemainingPower = InitialPower;
 }

public void turnRight (  )

 { ... }

public void turnLeft (  )

 { ... }

public void fire (  )

 { ... }
}
```

Atividades de programação concorrente

- **Processamento paralelo (ou concorrente):** execução simultânea de vários processos
  - Verdadeiro processamento paralelo requer várias CPUs
  - Pode ser simulada usando o time-sharing com uma única CPU