Introdução à Programação Orientada a Objetos e aos Algoritmos

Prof. Bruno Nogueira

Programação Orientada a Objetos

- Java é uma linguagem de programação orientada a objetos (POO). Mas o que isso significa?
  - O mundo ao redor é composto por objetos: pessoas, carros, prédios, ...
  - Cada um desses objetos é capaz de realizar algumas ações que impactam sobre outros objetos
  - Essa visão é adotada por POO: objetos que podem agir sozinhos ou interagir entre si
  - Um objeto em um programa representa uma **abstração** de um objeto do mundo real

Exemplo

- Imagine um programa para simular o tráfego em uma rodovia
  - Um objeto para representar cada carro
  - Um objeto para cada faixa da rodovia
  - Um objeto para cada sinalização
  - Um objeto para cada semáforo...

- A interação entre esses objetos nos permite simular o tráfego da rodovia

Terminologia - POO

- Em POO, as características de um objeto são chamados de **atributos**
  - Um carro, por exemplo, tem seu nome, sua velocidade atual e seu nível de combustível
  - Conjunto dos valores dos atributos dá ao objeto o seu **estado**
- As ações que um objeto podem tomar formam o seu comportamento
  - Em POO, um comportamento é definido por um conjunto de código, chamado **método**
  - Ações são acionadas por meio da invocação de métodos
    - Exemplos:
      - System.out.**println**("Olá.");
      - n1 = keyboard.**nextInt**();

Terminologia - POO

- Objetos de um mesmo tipo, têm os mesmos tipos de dados e o mesmo comportamento
  - Neste caso, dizemos que eles pertencem à mesma **classe**
  - Uma classe define o tipo de um objeto; é um diagrama, um esboço para a criação de objetos
  - No exemplo da simulação da rodovia, todos os automóveis pertencem à mesma classe – provavelmente chamada Automóvel
  - Todos os objetos de uma mesma classe têm os mesmos atributos e os mesmos métodos

Algoritmos

- Objetos têm comportamentos definidos pelos métodos
- É de responsabilidade do desenvolvedor criar estes métodos com os comportamentos corretos
  - Computador não tem senso próprio, deve receber instruções explícitas
  - A parte difícil não é colocar essa solução em uma linguagem de programação, mas sim desenvolver uma estratégia correta para desenvolver essas ações e comportamentos
  - Essa estratégia para criação de ações é expressa por meio de **algoritmos**

Algoritmos

- Um **algoritmo** é um conjunto de direções para resolver um problema
  - É como uma receita
  - Essas direções devem ser expressas de maneira muito completa e clara, tal que qualquer pessoa possa replicá-la
  - Não pode ser ambíguo
    - Por exemplo, um passo de um algoritmo: "Converta a leitura da temperatura de Celsius para Fahrenheit"
      - É uma instrução clara para um especialista, mas ambígua para um leigo
      - O problema está na representação, não no algoritmo

Algoritmos

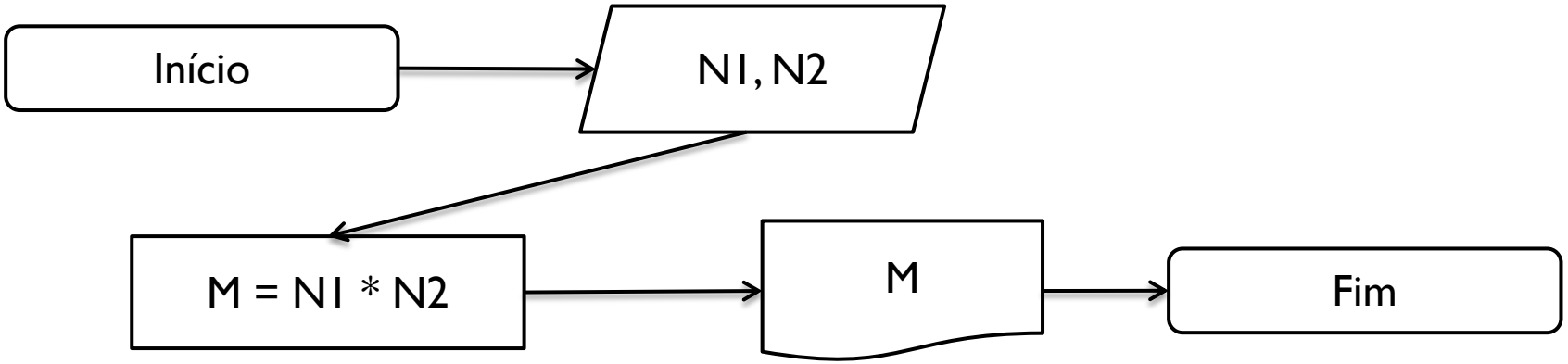
- Algoritmo deve possuir 3 qualidades
  1. Cada passo do algoritmo deve ser uma instrução que possa ser realizada
  2. A ordem dos passos deve ser precisamente definida
  3. O algoritmo deve ter fim
- Exemplo: algoritmo para o cálculo do custo total de uma lista de itens
  1. Escreva o número 0 em um quadro
  2. Para cada item da lista, faça o seguinte:
     1. Adicione o custo do item ao valor que está no quadro
     2. Substitua o número que está no quadro pelo resultado da soma
  3. Anuncie que a resposta é o número que está no quadro

Representação de algoritmos

- Descrição narrativa
  - Analisar o enunciado e escrever, em linguagem natural, os passos a serem seguidos na solução
    - Não é necessário aprender novos conceitos ☺
    - Abre espaço para várias interpretações, dificultando a transcrição para um programa ☹
  - Exemplo: algoritmo para exibir o resultado da multiplicação de dois números
    1. Receber os dois números que serão multiplicados
    2. Multiplicar os números
    3. Mostrar o resultado obtido da multiplicação

Representação de algoritmos

- Fluxograma
  - Analisar o enunciado e escrever, usando símbolos gráficos, os passos a serem seguidos na sua solução
    - Entendimento é mais simples que o entendimento de textos ☺
    - Fluxogramas devem ser entendidos e o algoritmo resultante não é detalhado, dificultando a transcrição para um programa ☹
  - Exemplo da multiplicação de dois números



Representação de algoritmos

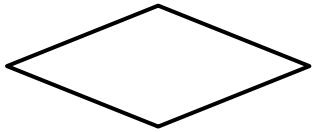
Início e fim do algoritmo

Fluxo de dados

Cálculos e atribuições de valores

Entrada de dados

Saída de dados



Tomada de decisão, com possíveis desvios

Teste de repetição

Representação de algoritmos

- Pseudocódigo
  - Analisar o enunciado e escrever, por meio de regras predefinidas, os passos a serem seguidos
  - É uma mistura de língua natural com linguagem de programação
    - A passagem do algoritmo para linguagem de programação é quase imediata ☺
    - As regras do pseudocódigo devem ser aprendidas ☹

Representação de algoritmos

- Exemplo da multiplicação de dois números

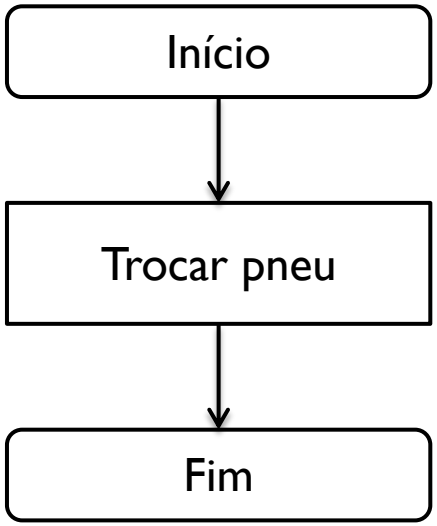
```
ALGORITMO

DECLARE N1, N2, M NUMÉRICO
ESCREVA "Digite dois números"
LEIA N1, N2
M ← N1 * N2
ESCREVA "Multiplicação =", M

FIM_ALGORITMO
```

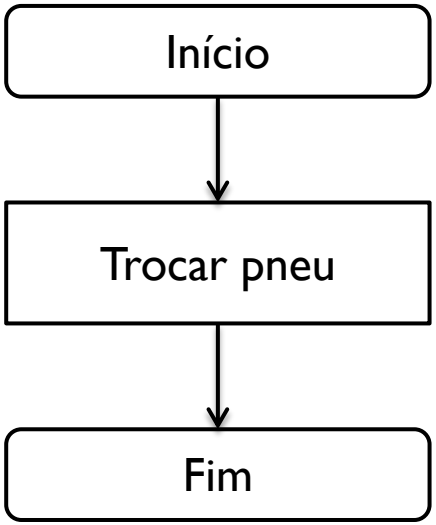
Estrutura de algoritmos

- Algoritmo para trocar pneu de um carro

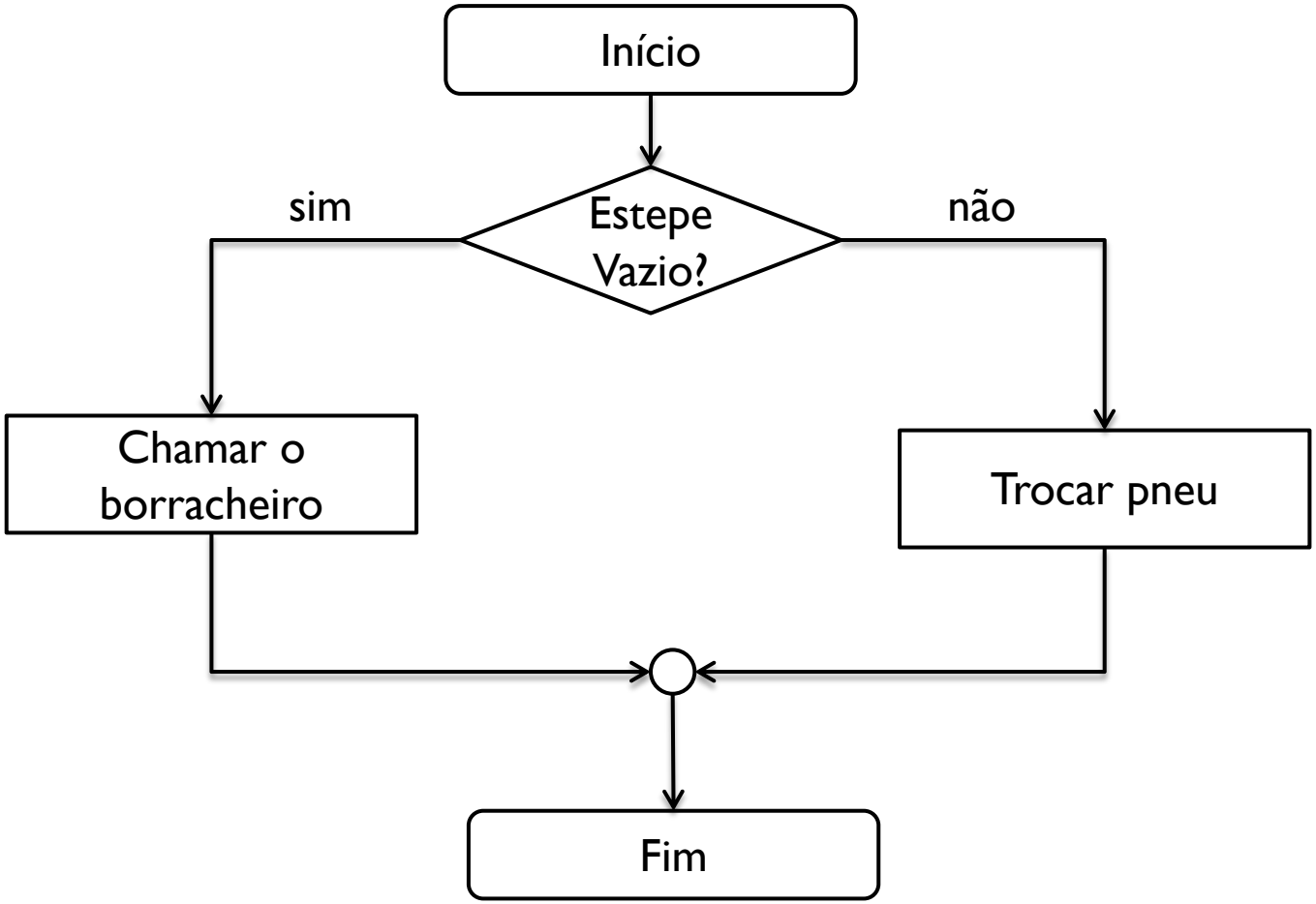


Estrutura de algoritmos

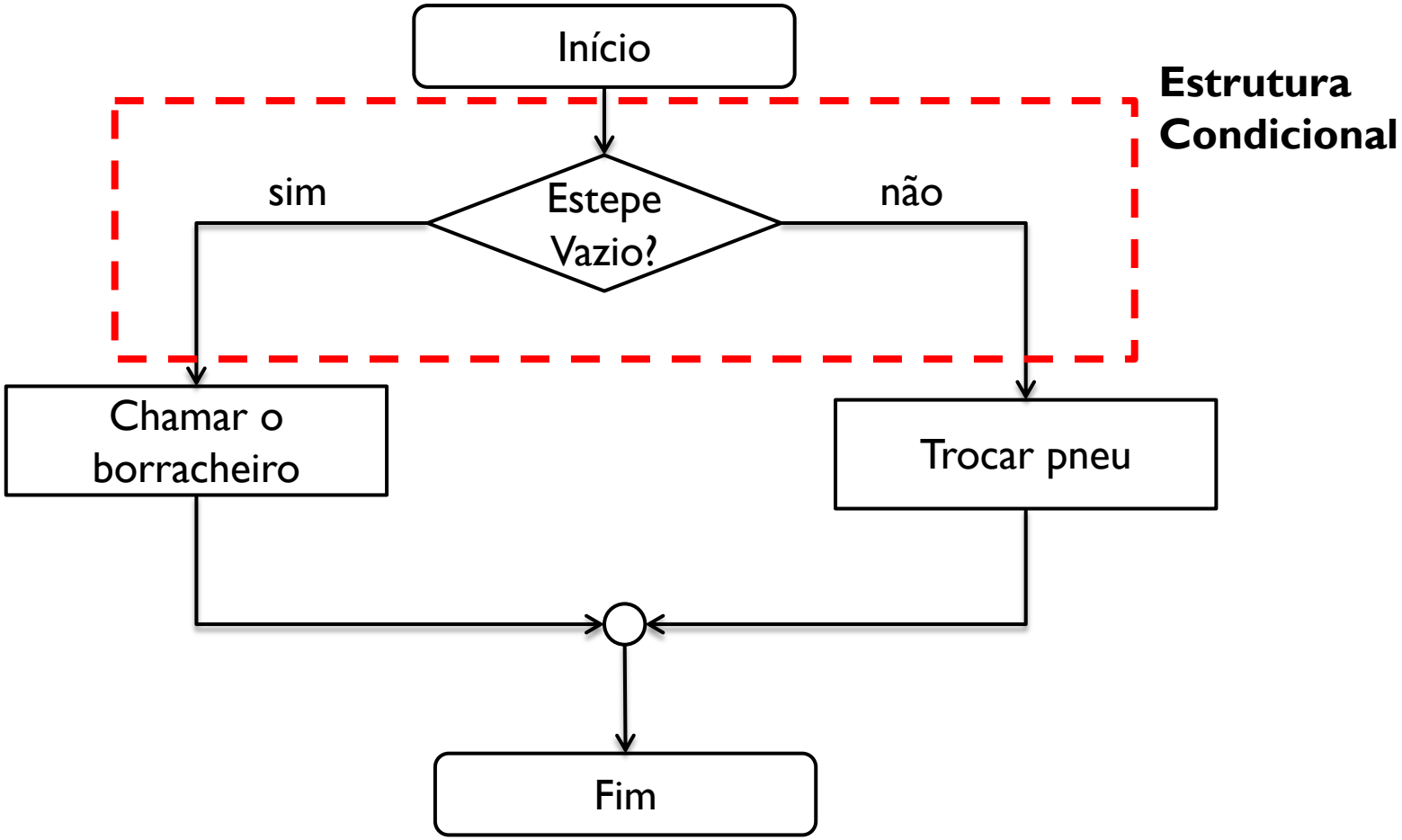
- Algoritmo para trocar pneu de um carro



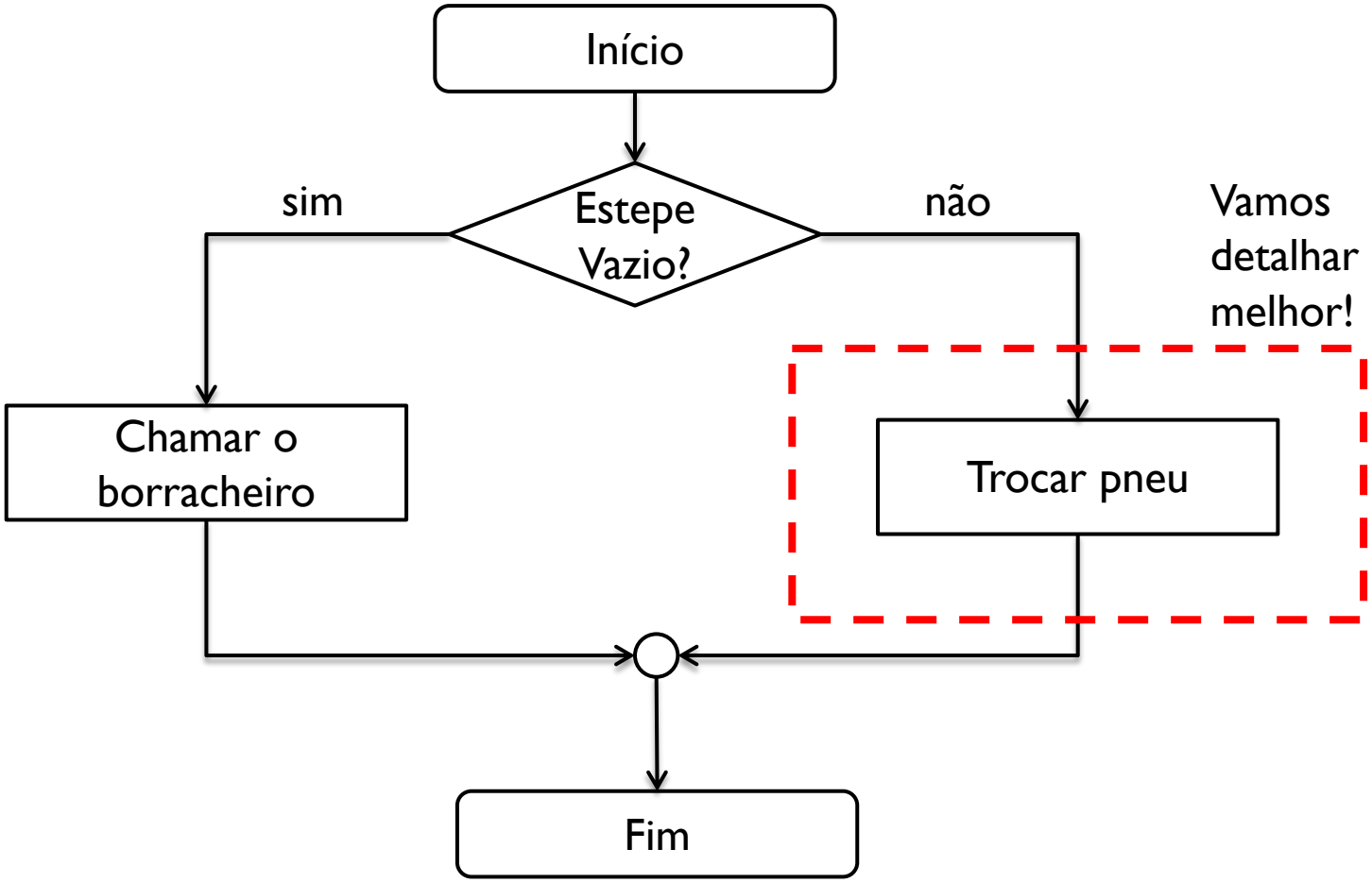
Estrutura de algoritmos



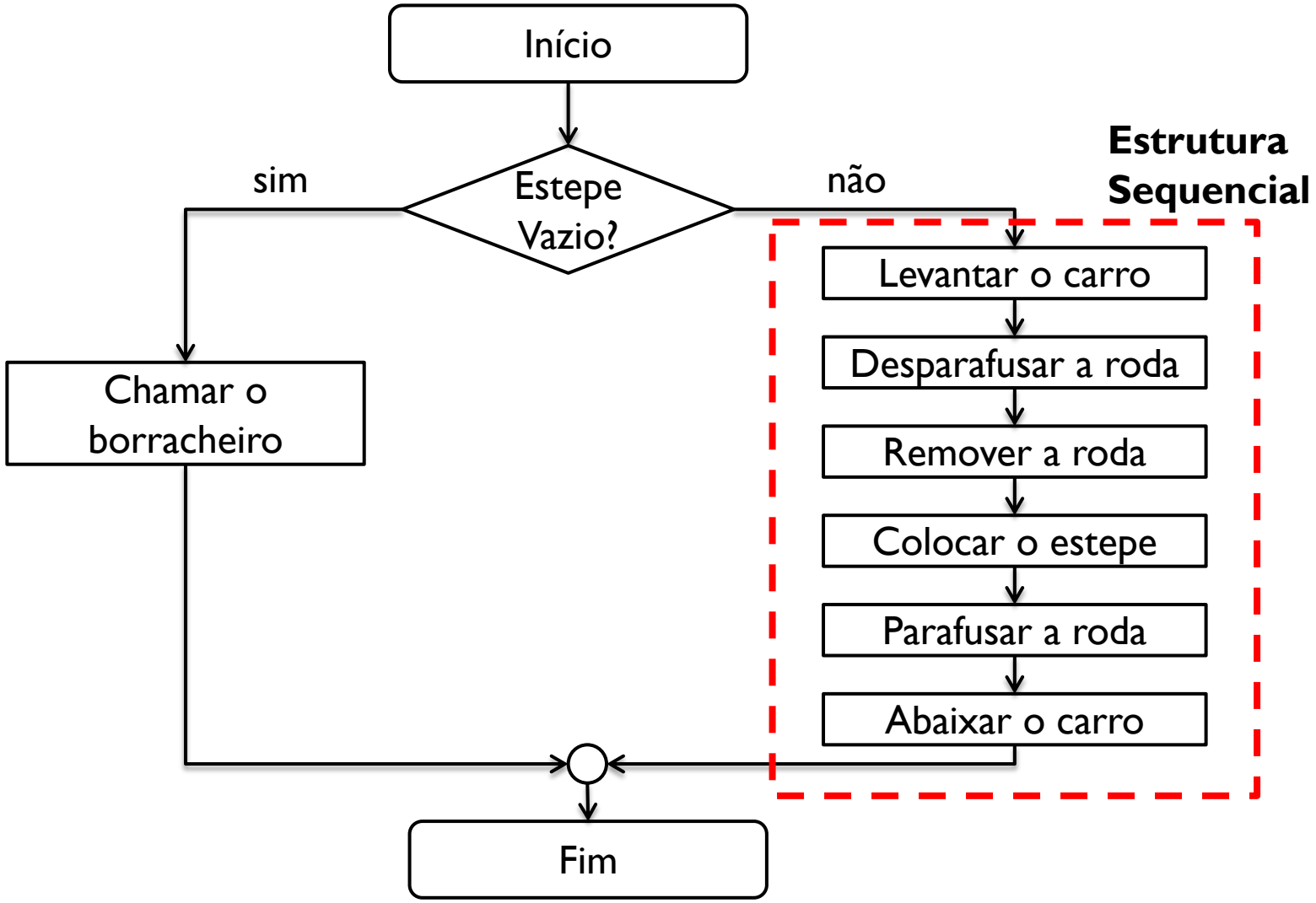
Estrutura de algoritmos



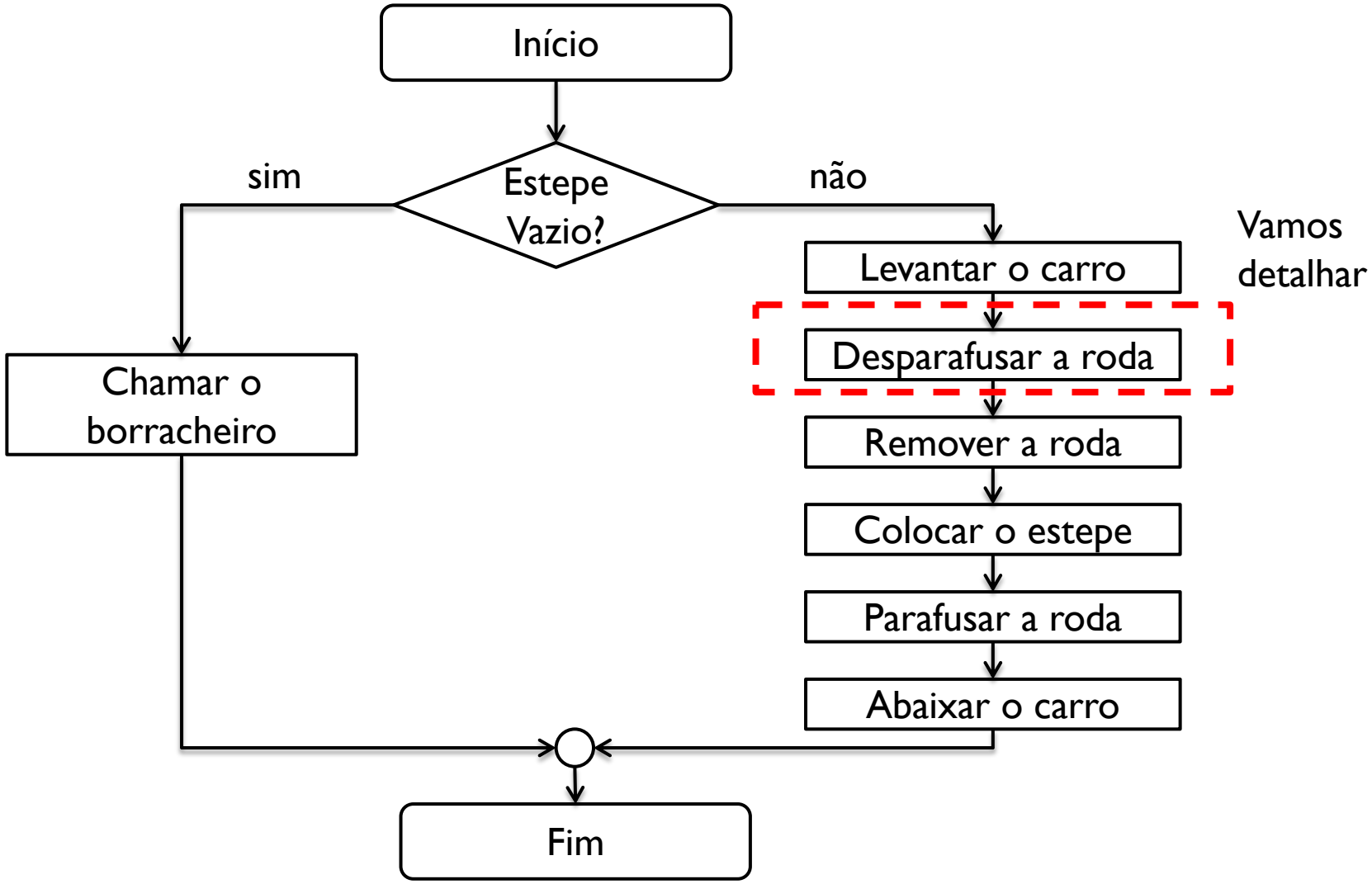
Estrutura de algoritmos



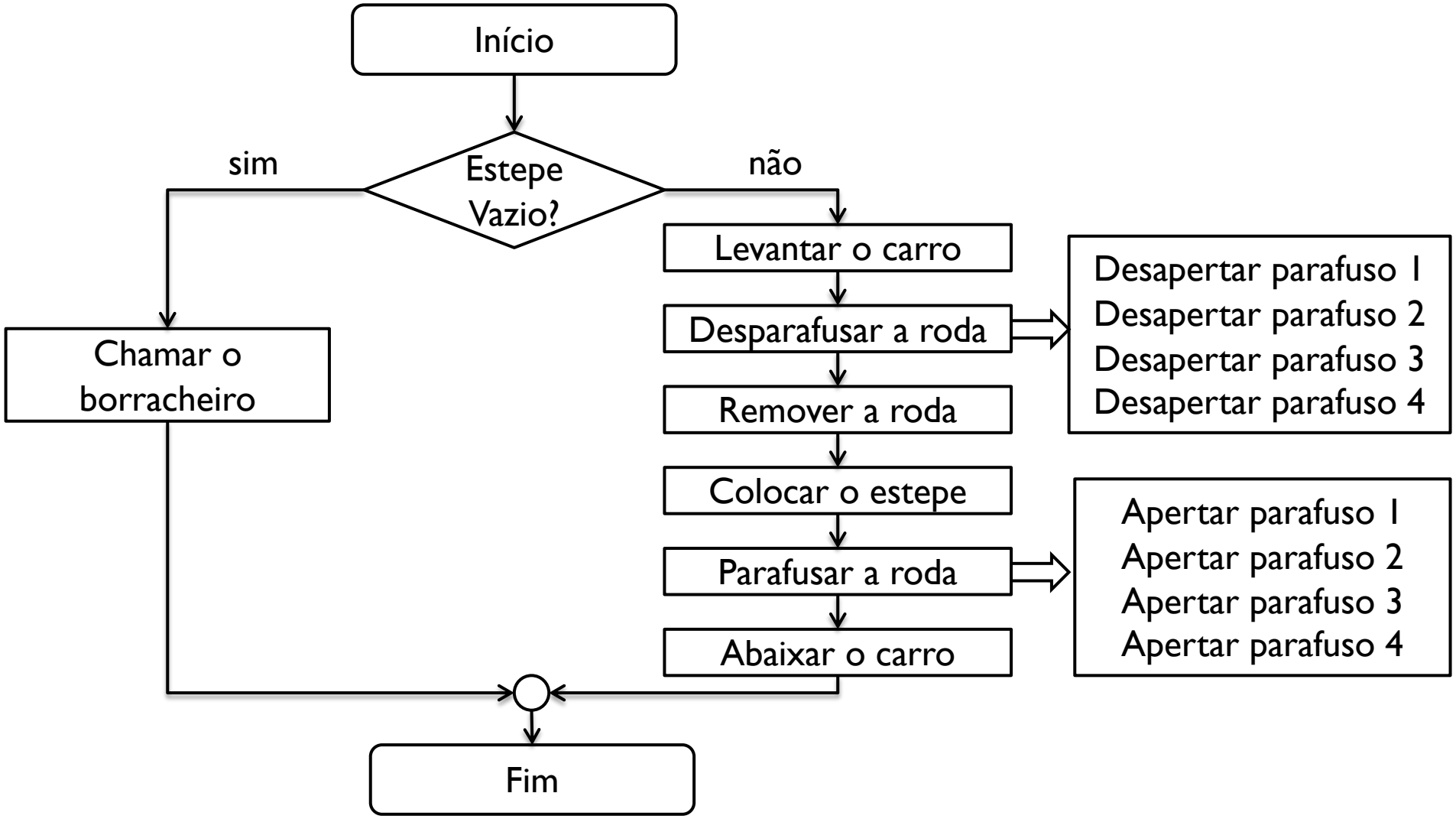
Estrutura de algoritmos



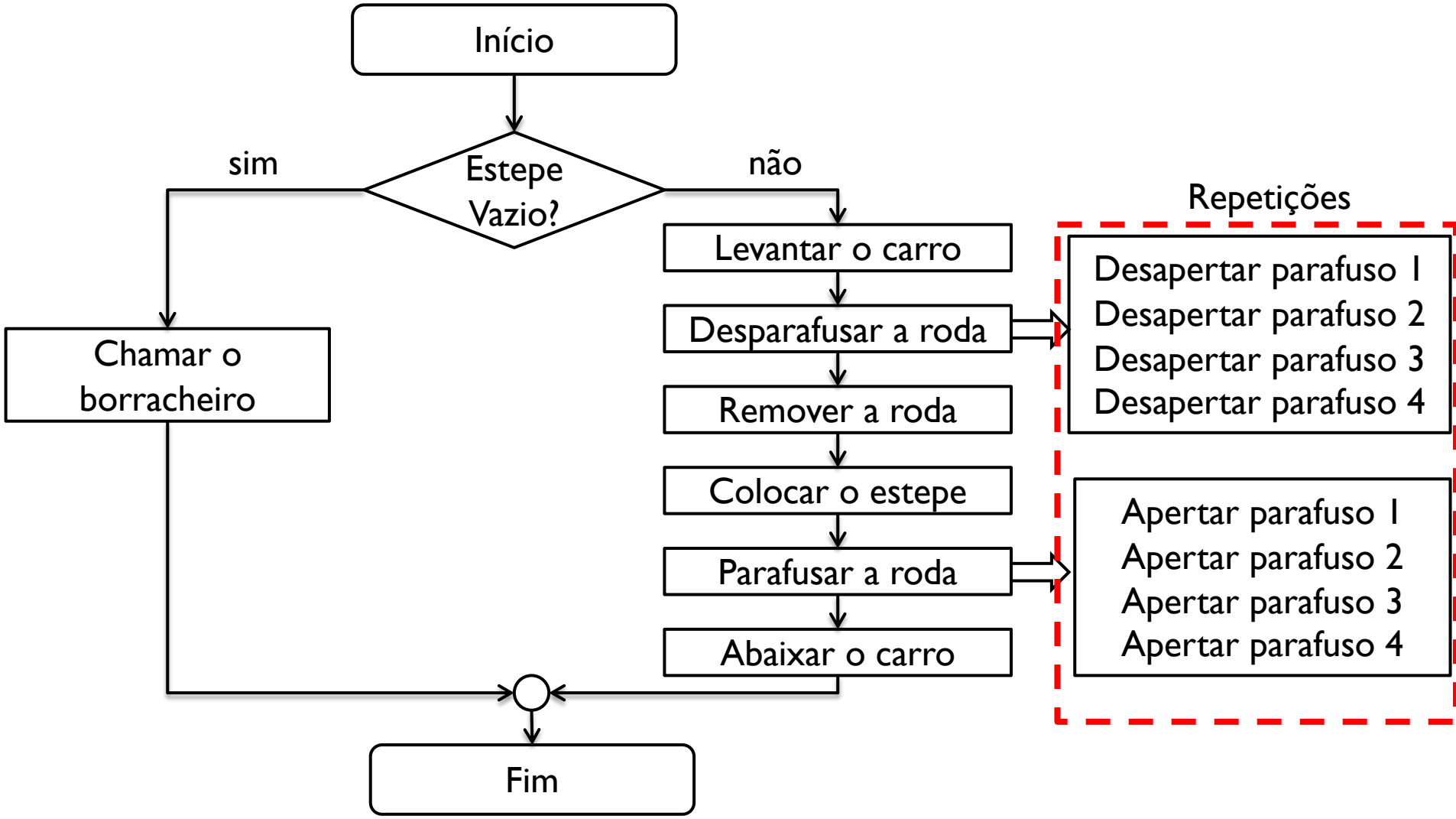
Estrutura de algoritmos



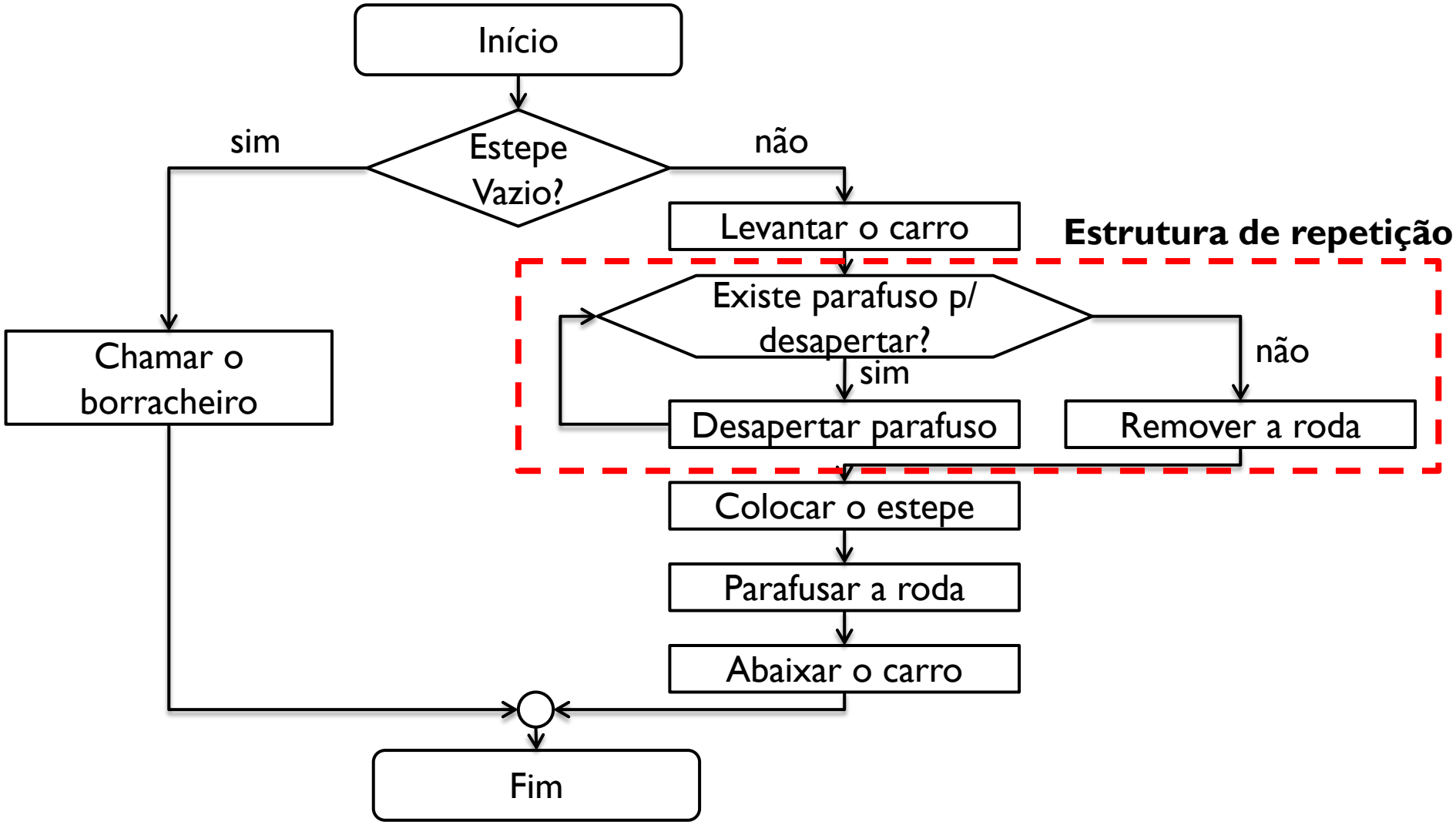
Estrutura de algoritmos



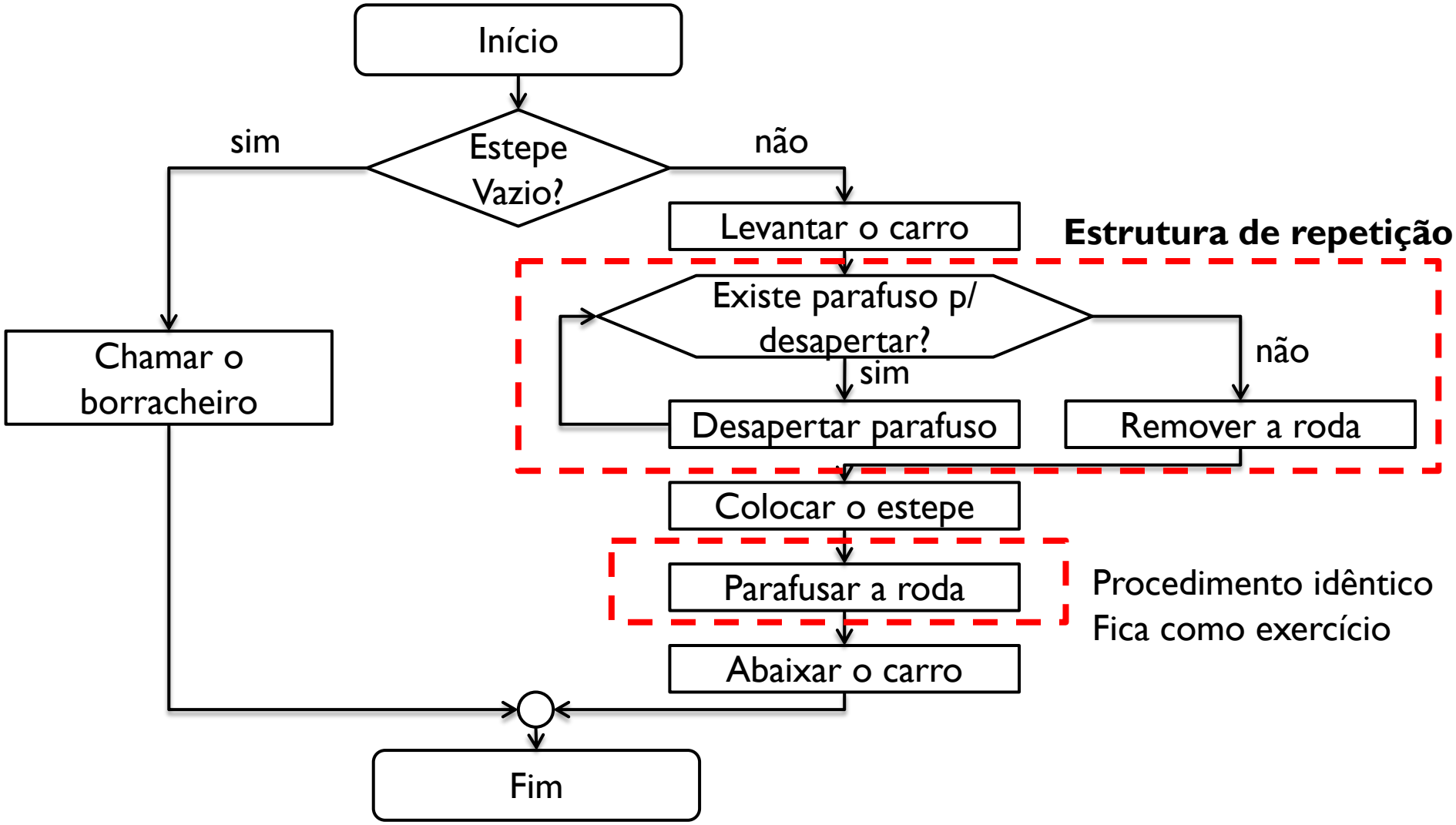
Estrutura de algoritmos



Estrutura de algoritmos

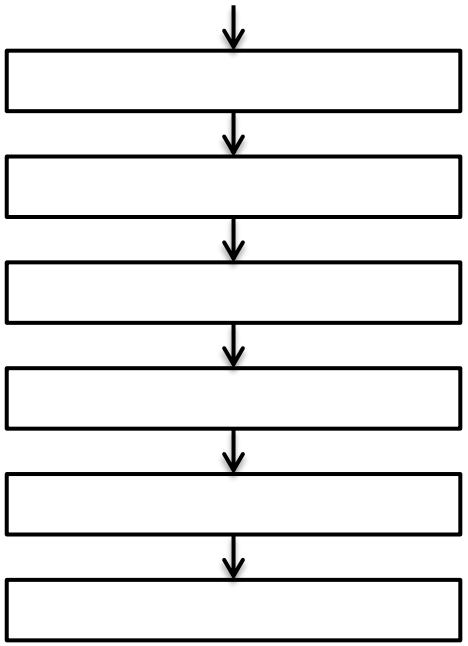


Estrutura de algoritmos



Estrutura de algoritmos

- **Estrutura sequencial**: passos são tomados em uma sequência predefinida



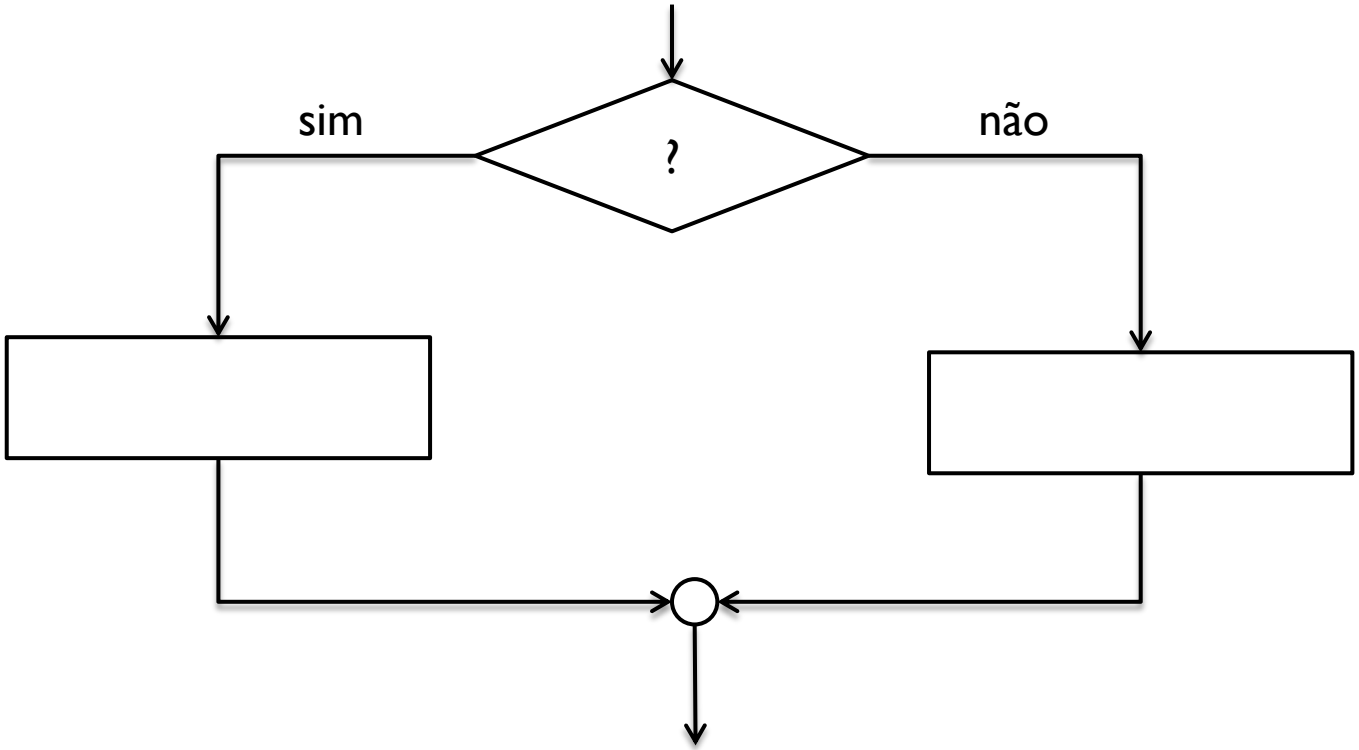
```
ALGORITMO

DECLARE N1, N2, M NUMÉRICO
ESCREVA "Digite dois números"
LEIA N1, N2
M ← N1 * N2
ESCREVA "Multiplicação =", M

FIM_ALGORITMO
```

Estrutura de algoritmos

- **Estrutura condicional**: permite a escolha do grupo de ações a ser executado quando uma condição é ou não satisfeita



Estrutura de algoritmos

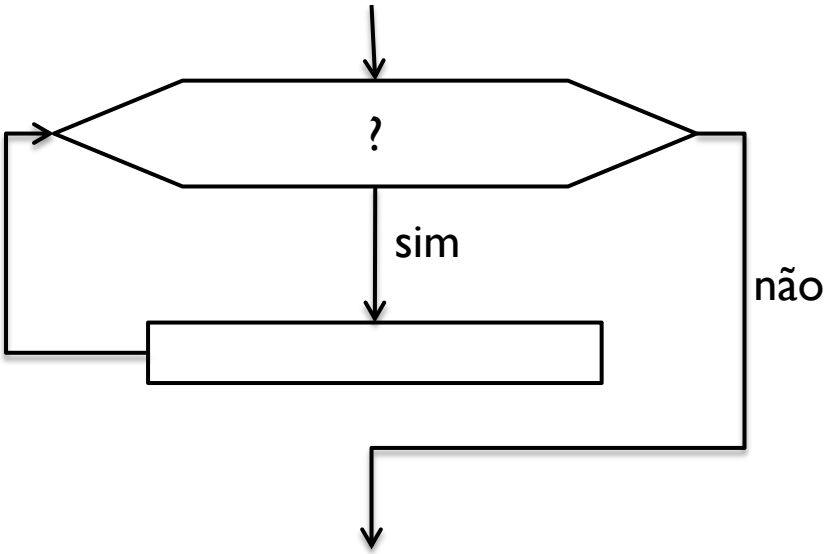
- **Estrutura condicional**: permite a escolha do grupo de ações a ser executado quando uma condição é ou não satisfeita

```
ALGORITMO

DECLARE N1, N2 NUMÉRICO
ESCREVA "Digite dois números"
LEIA N1, N2
SE N1 < N2 FAÇA
    ESCREVA "N1 é menor que N2"
FIM_SE
SENÃO FAÇA
    ESCREVA "N1 é maior ou igual a N2"
FIM_SENÃO
FIM_ALGORITMO
```

Estrutura de algoritmos

- **Estrutura de repetição**: permite que uma sequência de comandos seja executada repetidamente até que uma determinada condição de parada seja satisfeita



Estrutura de algoritmos

- **Estrutura de repetição**: permite que uma sequência de comandos seja executada repetidamente até que uma determinada condição de parada seja satisfeita

```
ALGORITMO

DECLARE CONT NUMÉRICO
CONT = 10
ENQUANTO CONT > 0 FAÇA
   ESCREVA "Contador = ", CONT
   CONT = CONT – 1
FIM_ENQUANTO
FIM_ALGORITMO
```

Exercícios

Resolva os exercícios utilizando a representação por fluxograma e pseudocódigo

1. Elaborar um algoritmo que calcule a área de um triângulo retângulo
2. Faça um algoritmo que exiba o resultado da divisão de dois números
3. Criar um algoritmo que resolva uma equação do segundo grau na forma:

$$Ax^2 + Bx + C = 0$$

sendo que A, B e C são fornecidos pelo usuário. Considere que A, B e C são números reais e A é diferente de 0.