

SSC0304 - Introdução à Programação para Engenharias

# Conceitos de Computação: Algoritmos e Programação

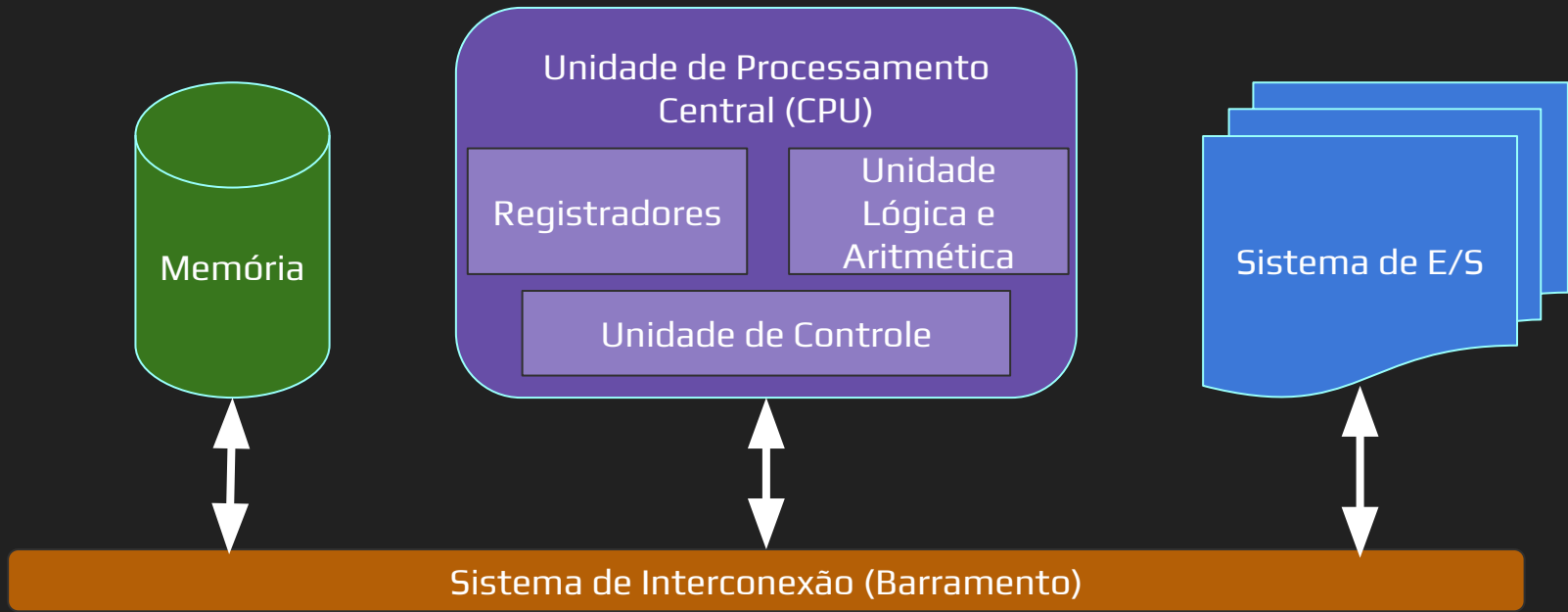
Prof.: Leonardo Tórtoro Pereira

[leonardop@usp.br](mailto:leonardop@usp.br)

Baseado no material dos profs Fernando S. Osório e Claudio F.M. Toledo

Na aula passada...

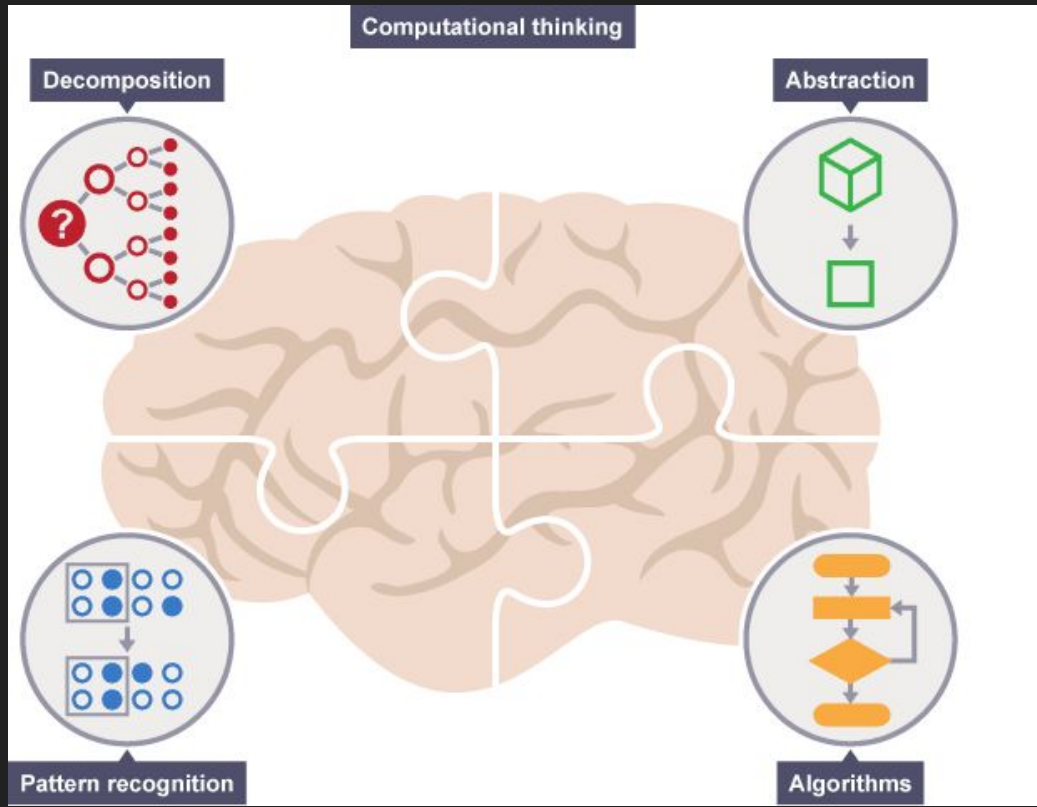




Arquitetura de von Neumann. Fonte: [1, 2]



Pirâmide DIKW. Fonte: [3]



Pensamento computacional. Fonte: [5]

O que vamos aprender hoje?



# Objetivos

- Entender o conceito de algoritmos e como representá-los através de fluxogramas
- Aprender sobre os elementos de um fluxograma
- Aprender sobre os diferentes tipos e paradigmas de linguagens de programação

# Tópicos da Aula

- Algoritmos e Fluxogramas
- Tipos e variações entre linguagens de programação



# Algoritmos e Fluxogramas

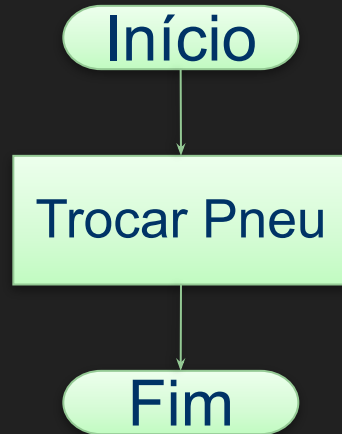
# Algoritmos

- Computador não pensa sozinho
  - ◆ Precisa receber instruções explícitas
  - ◆ Algoritmos
    - Sequência de instruções

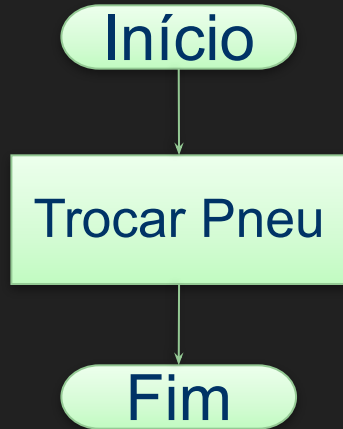
# Algoritmos

- Precisa ter 4 qualidades
  - ◆ Cada passo do algoritmo deve ser uma instrução que possa ser realizada (codificada no computador)
  - ◆ A ordem dos passos deve ser precisamente determinada
  - ◆ O algoritmo deve ter fim (terminar)
  - ◆ O algoritmo deve ter um fim (uma utilidade/um objetivo)

# Algoritmo para trocar um pneu

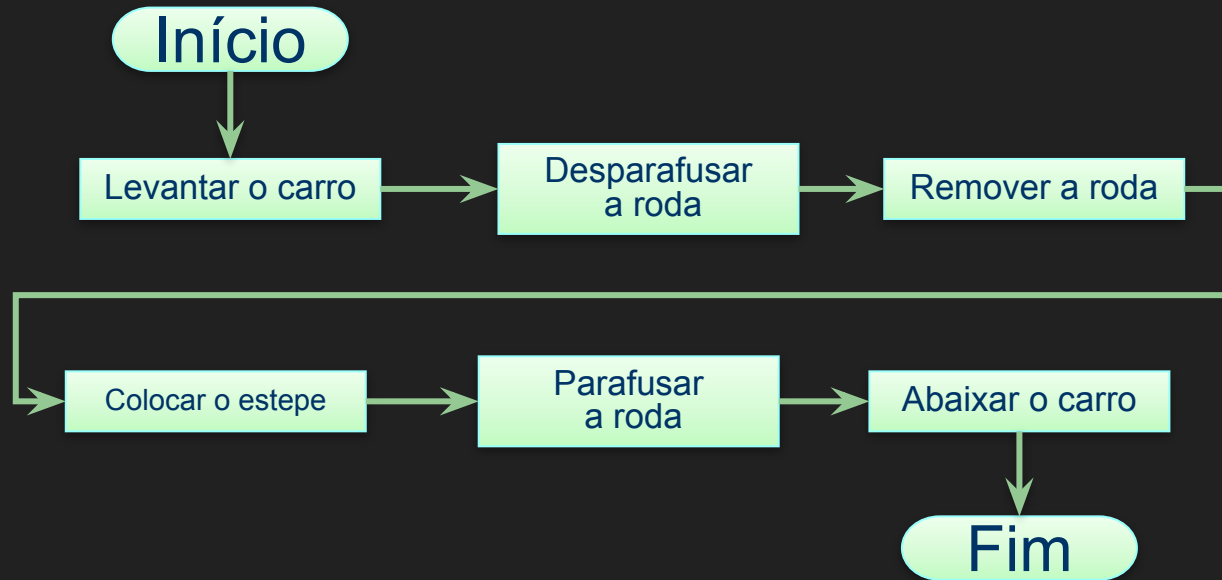


# Algoritmo para trocar um pneu

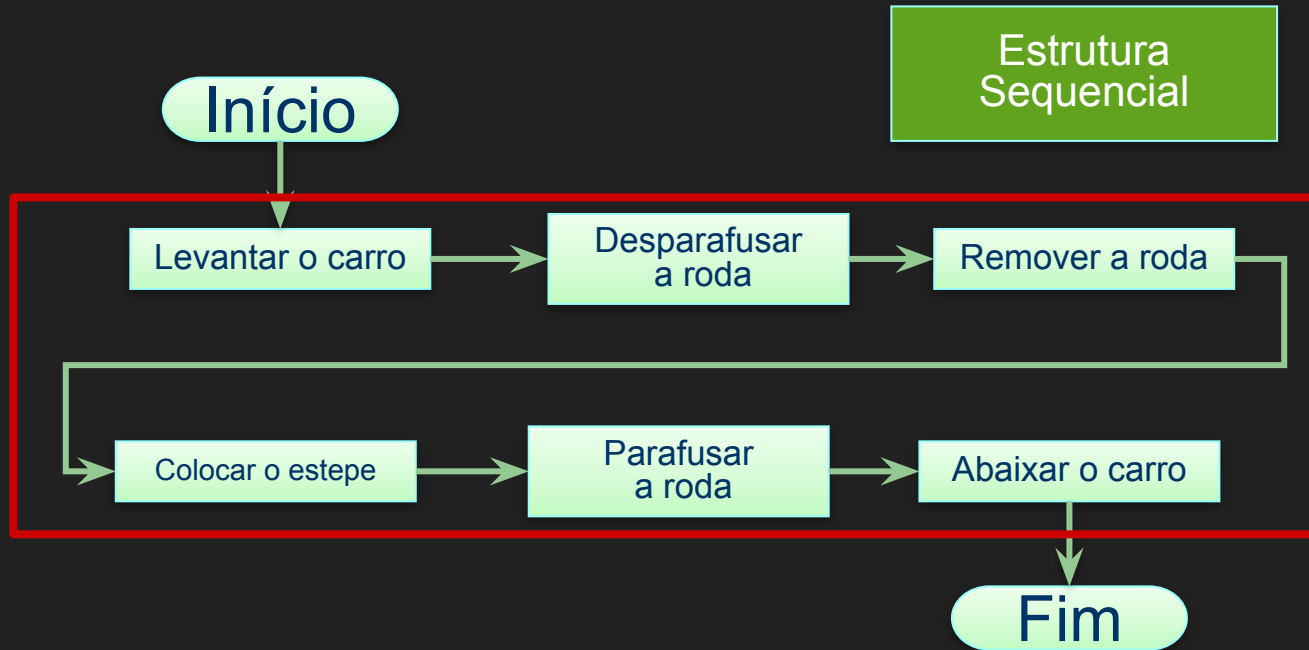


Trocar pneu?  
É suficientemente  
claro para você?

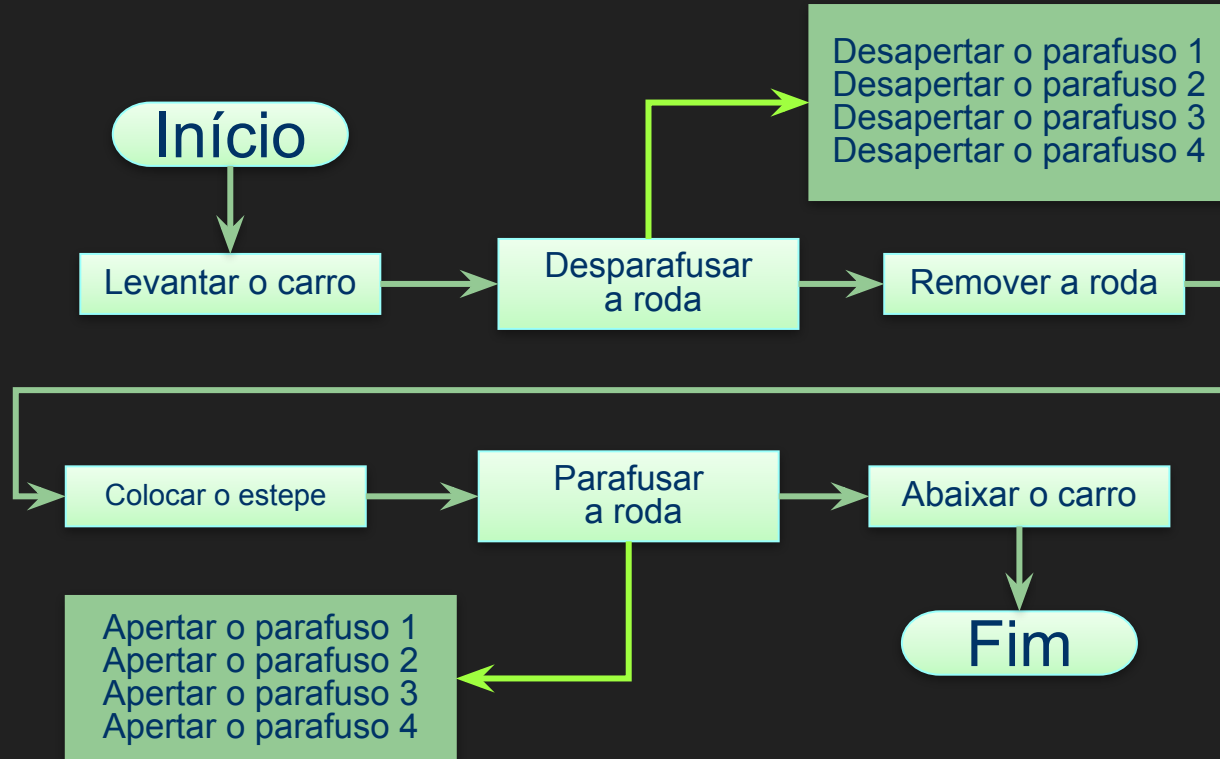
# Algoritmo para trocar um pneu



# Algoritmo para trocar um pneu



# Algoritmo para trocar um pneu





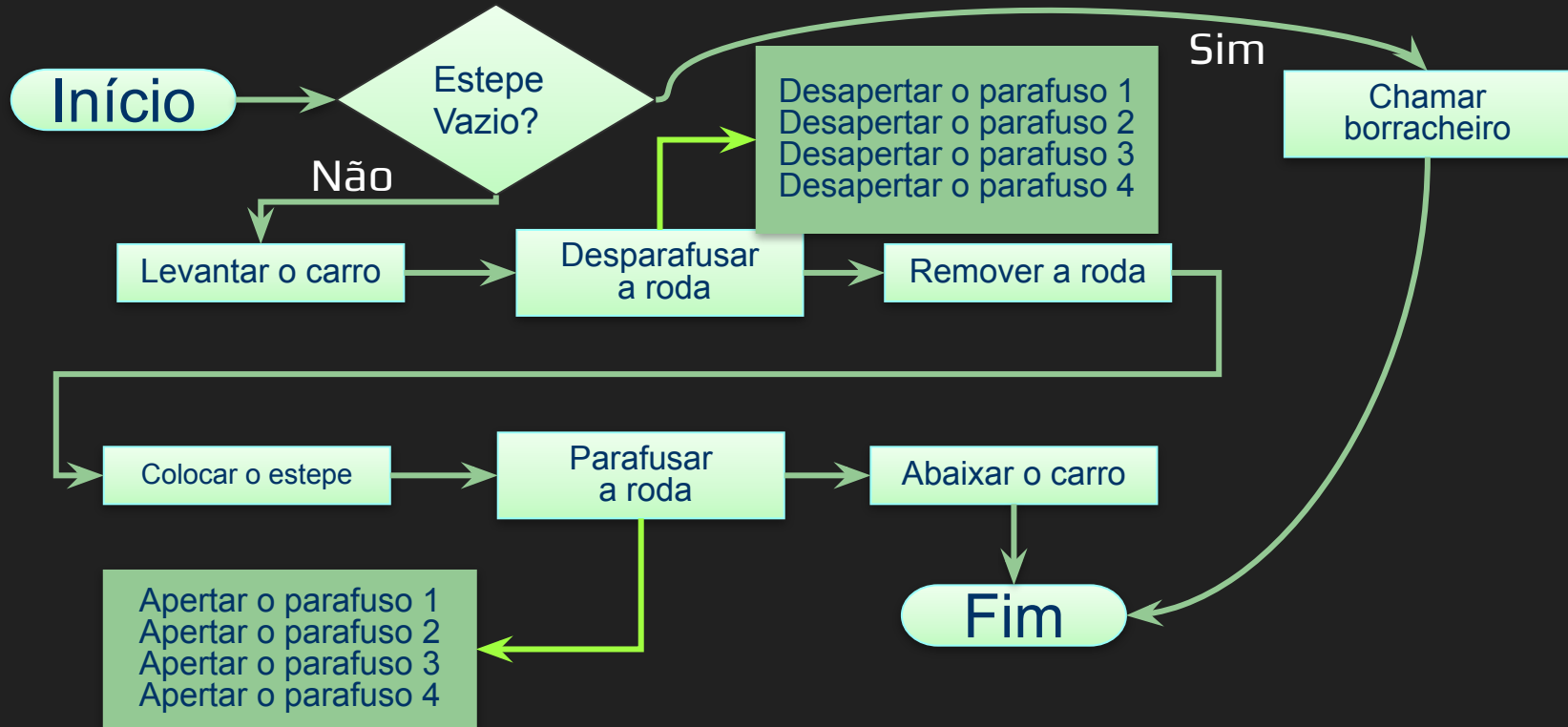
# Algoritmo para trocar um pneu



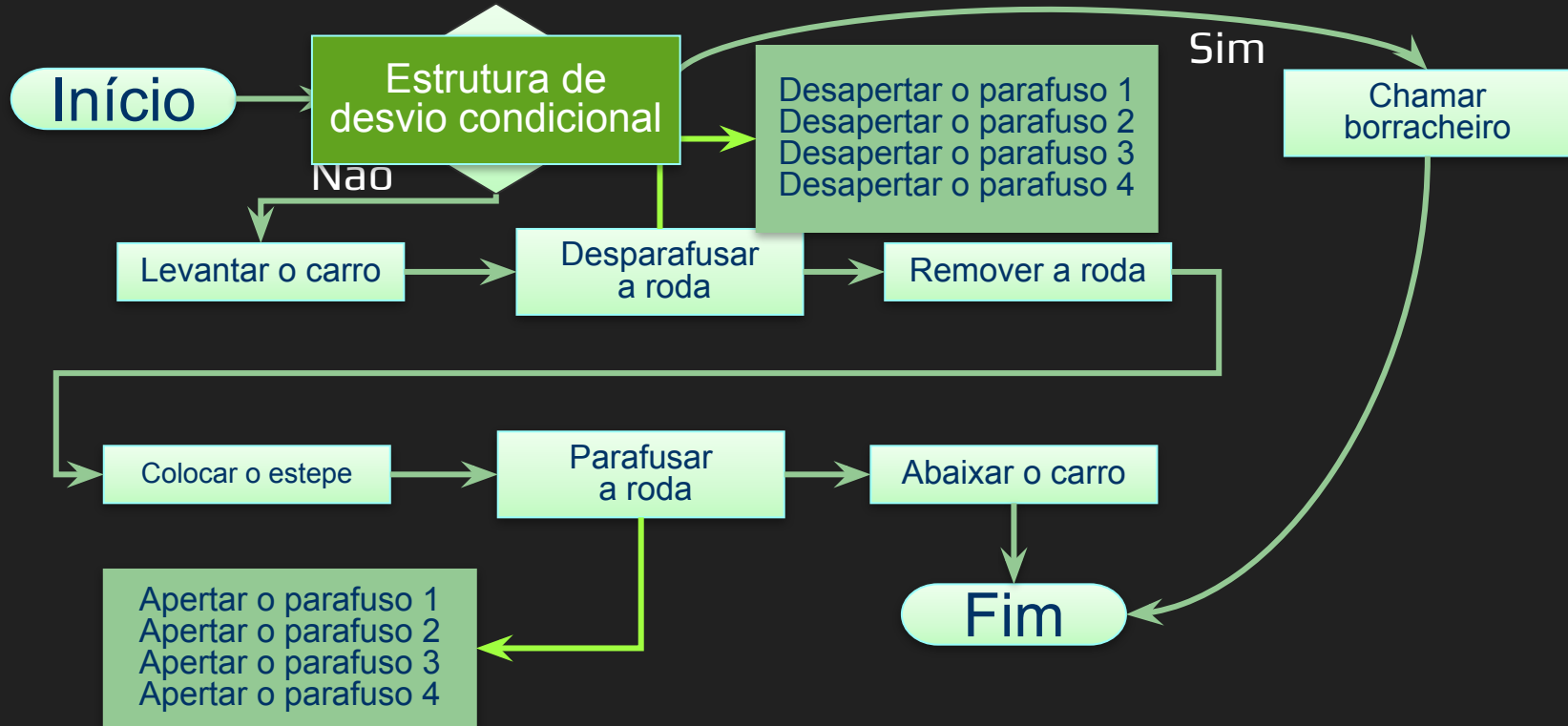
## E se...

- E se...
- Se não for possível seguir estes passos?
- Se algo não sair como previsto?
- Se eu tiver mais de uma alternativa?
- Se ...

# Algoritmo para trocar um pneu



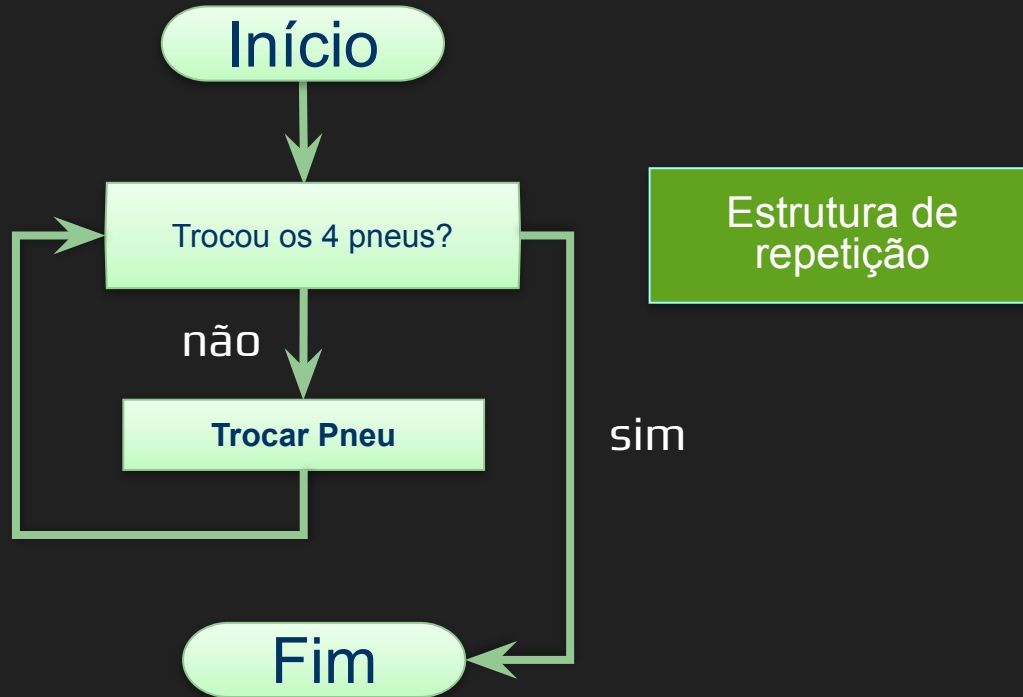
# Algoritmo para trocar um pneu



## E por que não... Pit Stop?

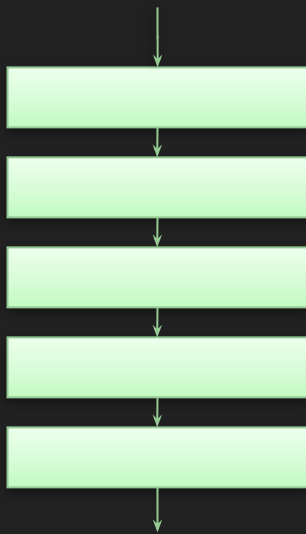
- E por que não... Pit Stop?
  - ◆ Trocar os 4 pneus do carro

# Algoritmo para trocar um pneu



# Estruturas dos Algoritmos

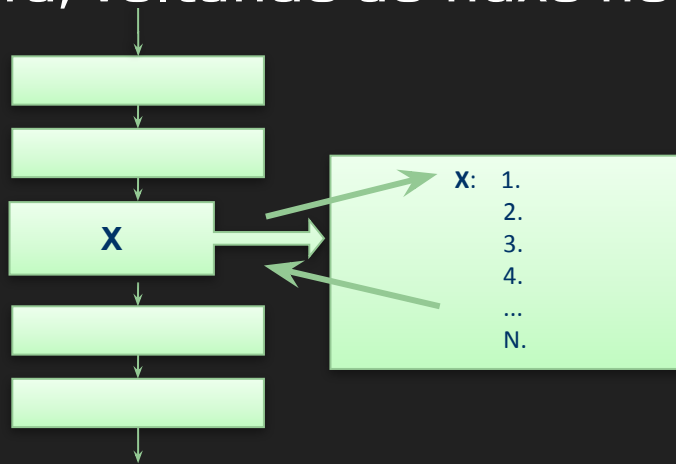
→ Em uma estrutura seqüencial, os passos são tomados em uma seqüência pré-definida.



Estrutura  
Seqüencial

# Estruturas dos Algoritmos

→ Em uma estrutura de sub-rotina, a execução é desviada para uma seqüência de comandos que executam uma tarefa, voltando ao fluxo normal

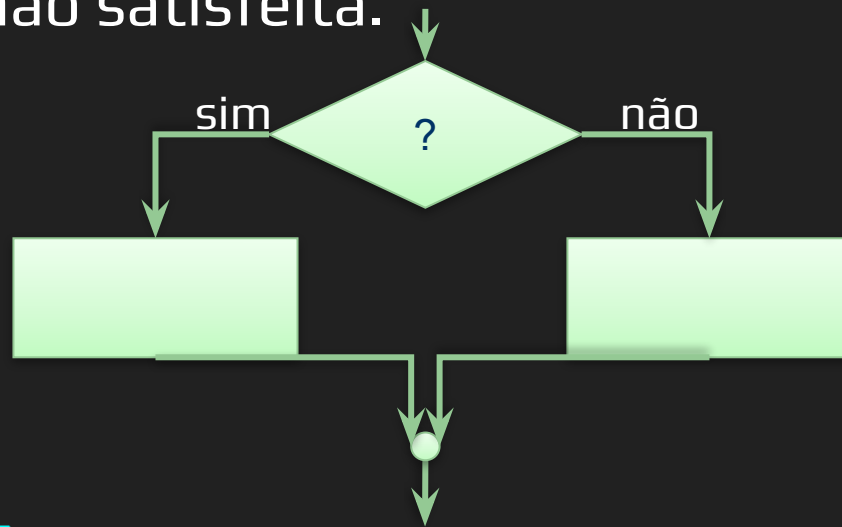


Estrutura de  
Sub-Rotina



# Estruturas dos Algoritmos

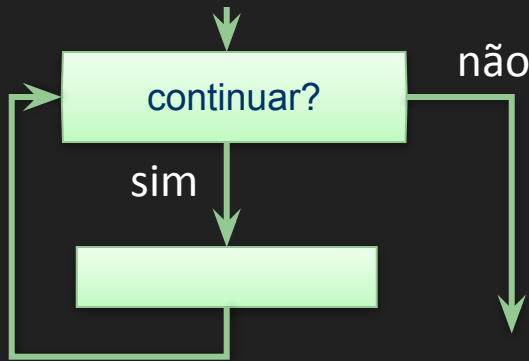
→ Uma estrutura condicional permite a escolha do grupo de ações a ser executado quando determinada condição é ou não satisfeita.



Estrutura  
Condicional

# Estruturas dos Algoritmos

- Uma estrutura de repetição permite que uma seqüência de comandos seja executada repetidamente até que uma determinada condição de interrupção seja satisfeita.



Estrutura de  
Repetição

## Exercício

1. Faça um algoritmo que descreva a preparação de um bolo de chocolate
2. Faça um algoritmo que descreva a preparação de uma dúzia de copos de suco de laranja

# Desafio

→ Resolva os problemas abaixo (ver sites abaixo)

◆ 4.1 Lobo, Ovelha e Alface -

<https://www.proprofsgames.com/wolf-sheep-and-cabbage/>

◆ 4.2 Missionários e Canibais -

<https://www.novelgames.com/en/missionaries/>

# Programas

# Algoritmos -> Programas

→ Computador:

- ◆ Uso de dados armazenados na memória (variáveis)
- ◆ Instruções bem definidas: os comandos da linguagem

→ Ciclo:

- ◆ Entrada de Dados: Ler os dados
- ◆ Processamento: manipular os dados
- ◆ Saída de Dados: Escrever os resultados

# Programa de Computador: Memória

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | ... | N |
|---|---|---|---|---|---|---|-----|---|
|   |   |   |   |   |   |   |     |   |
|   |   |   |   |   |   |   |     |   |

- A memória do computador armazena dados (bytes)
- Cada dado tem a sua posição na memória (endereço)

# Programa de Computador: Memória

| 1      | 2      | 3         | 4      | 5   | 6    | 7     | ... | N |
|--------|--------|-----------|--------|-----|------|-------|-----|---|
| Pregos | Porcas | Parafusos | Açúcar | Sal | Óleo | Leite |     |   |
|        |        |           |        |     |      |       |     |   |

- A memória do computador armazena dados (bytes)
- Cada endereço pode armazenar diferentes tipos de dados (variáveis)



# Programa de Computador: Memória

| 1      | ...   | 7    | ... | N    |    |        |
|--------|-------|------|-----|------|----|--------|
| Pregos | Fusos | usos | ar  | ite  |    |        |
| 30     | 45    | 45   | 2kg | 300g | 1L | 250 ml |

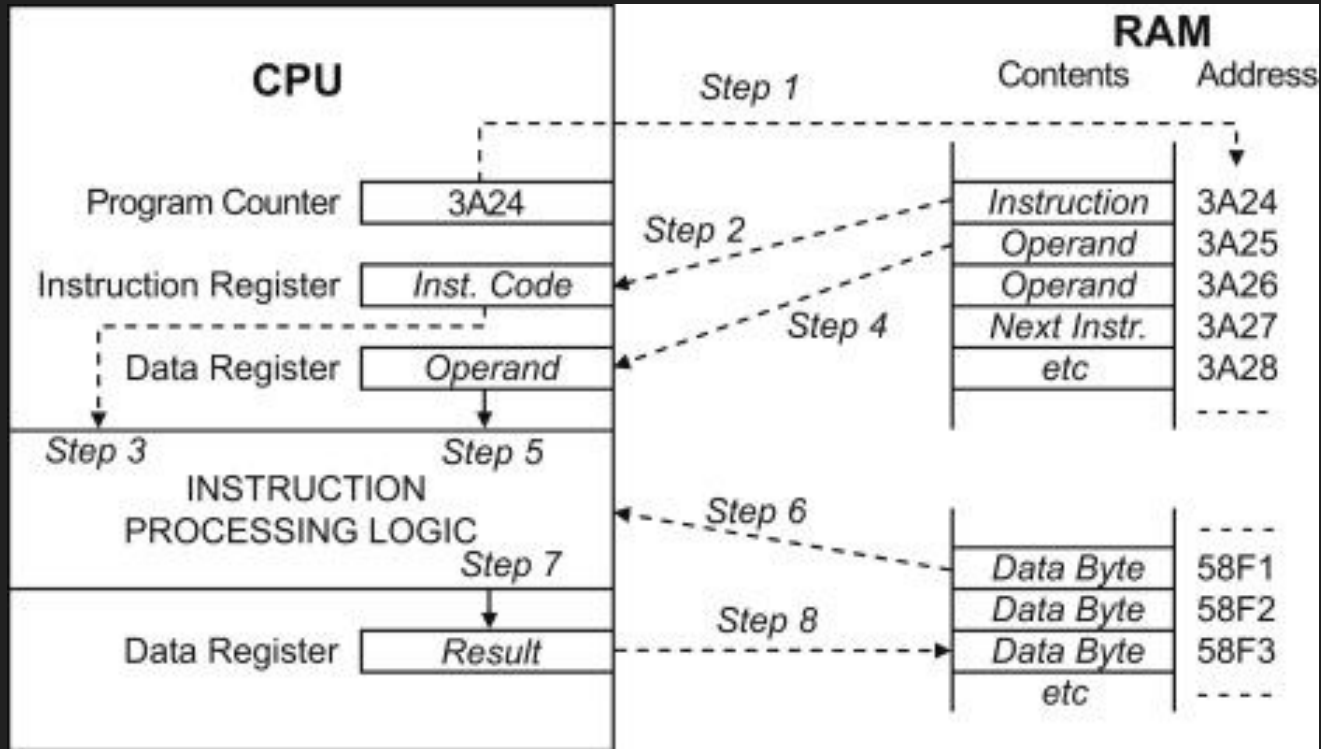
quantidade\_de\_pregos\_disponiveis

- Variáveis vão receber NOMES...
- Nomes que representam uma informação (valor armazenado) de um determinado tipo em uma determinada posição da memória

# Comandos

- Comandos são ordens para que o computador manipule os dados de sua memória...
- Exemplos de Comandos:
  - Realizar operações com os dados: mover, somar, subtrair
  - Ler novos dados pelo teclado: entrada de dados
  - Escrever resultados na tela: saída de dados

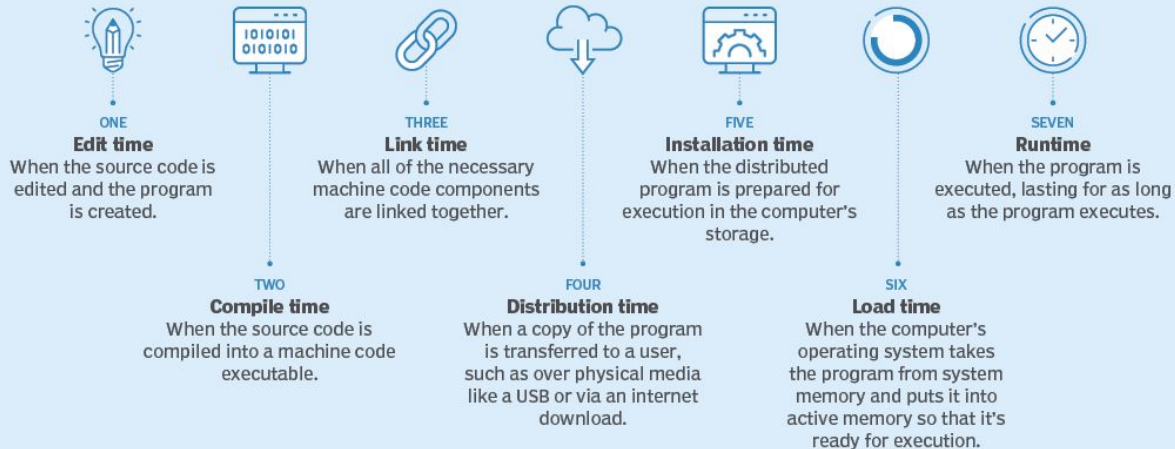
# Exemplo de comandos em Python



Fluxo de execução de um programa

<https://www.sciencedirect.com/topics/engineering/program-execution>

# The programming lifecycle

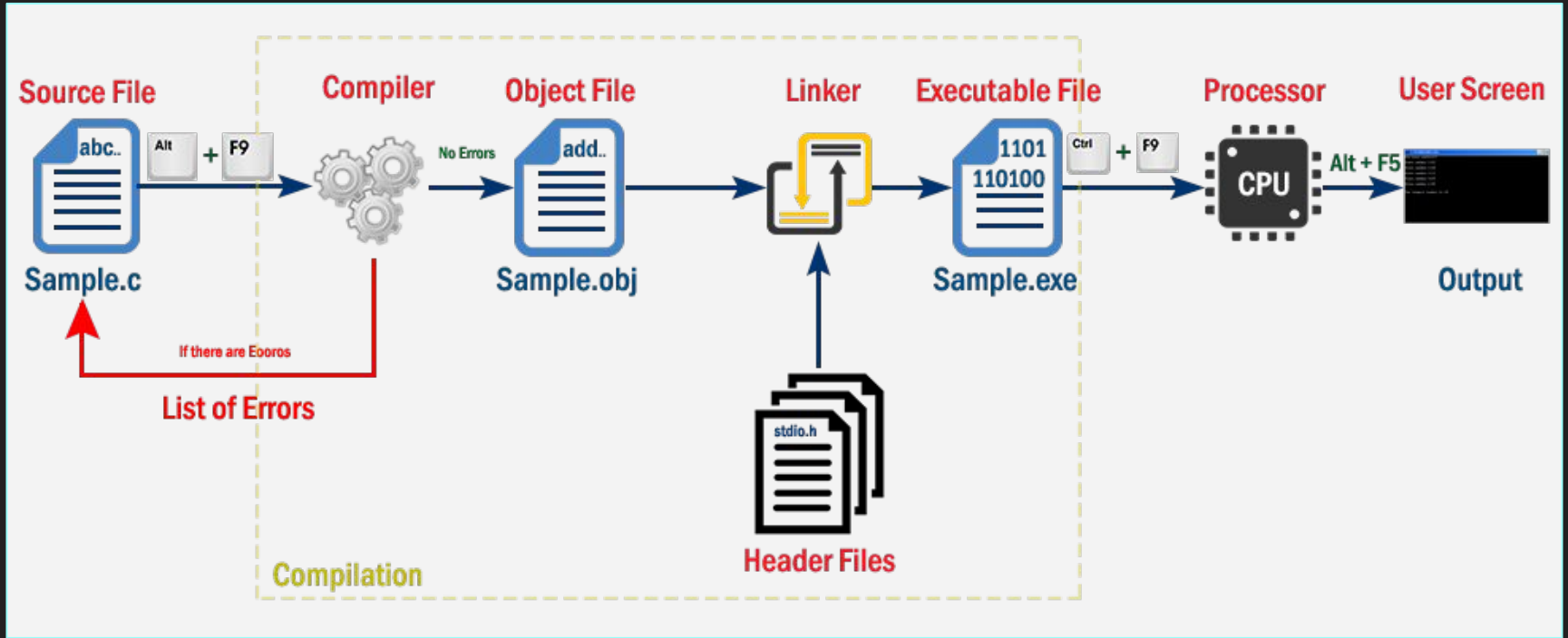


ICONS: DA-V O'DA, MOMENTO DESIGN, SHV ECTORR AND RANEEZ/GETTY IMAGES

©2022 TECHTARGET. ALL RIGHTS RESERVED 

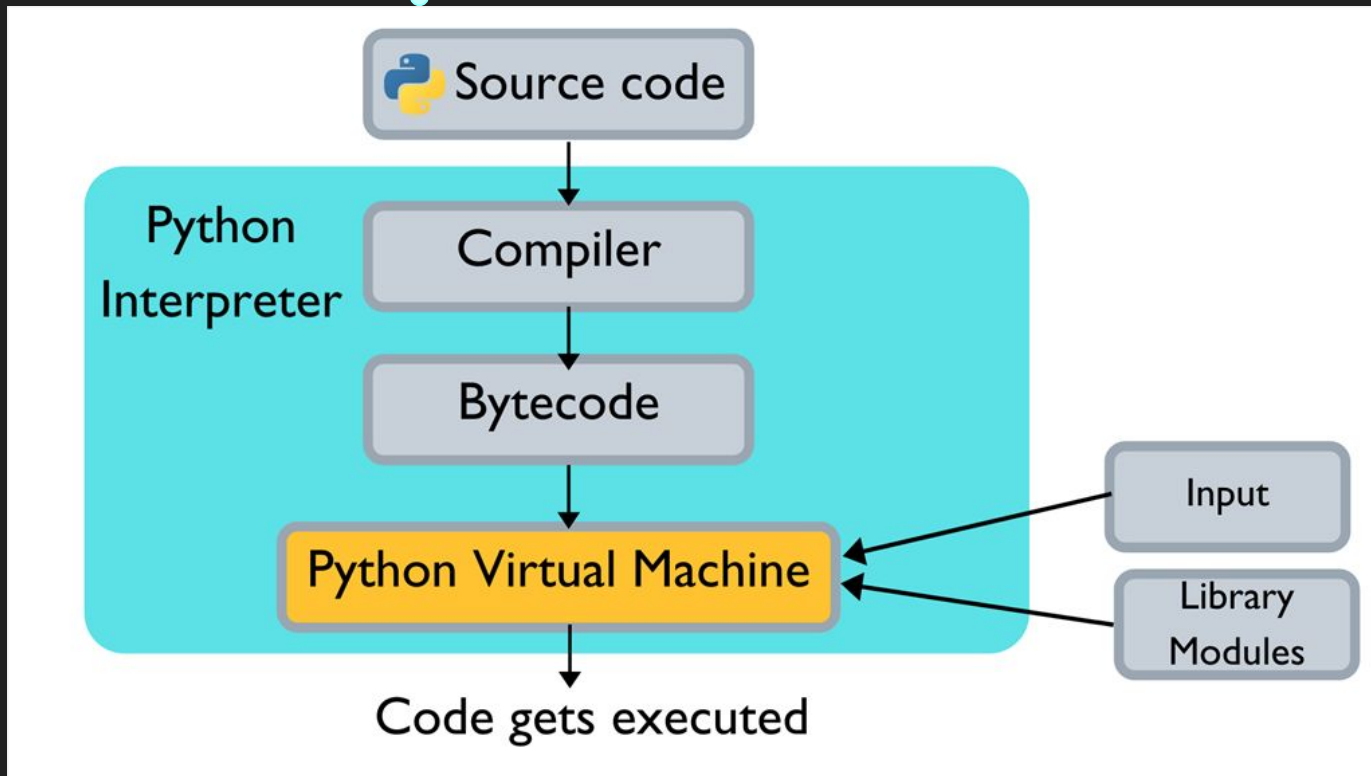
Ciclo de vida de um software

<https://www.techtarget.com/searchsoftwarequality/definition/runtime>



## Ciclo de vida de um programa em C

[http://www.btechsmartclass.com/c\\_programming/C-Creating-and-Running-C-Program.html](http://www.btechsmartclass.com/c_programming/C-Creating-and-Running-C-Program.html)



Ciclo de vida de um programa em Python

<https://www.c-sharpcorner.com/article/why-learn-python-an-introduction-to-python/>

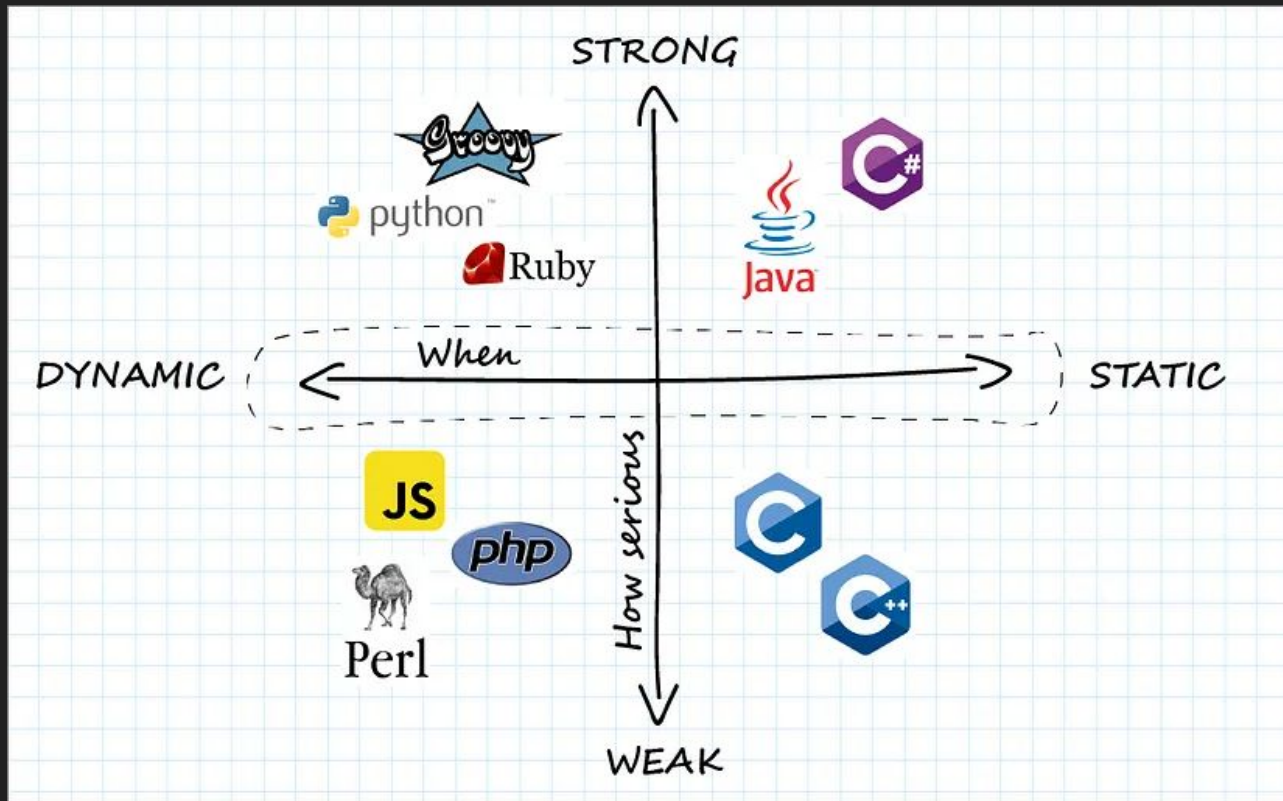


E qual linguagem usar?



# Tipagem

- Tipagem forte vs fraca
- Tipagem estática vs dinâmica



Comparação de linguagens por tipagem. Fonte [1]

# Tipagem forte

# Tipagem forte [1]

- Regras de tipagem rigorosas
- Tipos de dados precisam ser especificados
  - ◆ Não permitem conversão implícita entre tipos não relacionados
  - ◆ Ex: float não pode ser convertido para uma string

# Tipagem forte [1]

## → Vantagens

- ◆ Previne erros em tempo de execução
- ◆ Não existe atraso em tempo de execução para definição de tipos
- ◆ Código é melhor otimizado para o compilador
- ◆ Menos bugs, o que, no geral, agiliza o desenvolvimento

# Tipagem forte [1]

## → Desvantagens

- ◆ Perda de flexibilidade para programadores
- ◆ Variável só pode ser atribuída ao tipo designado
- ◆ Não é possível burlar as restrições impostas pelo sistema de tipagem

# Tipagem forte [1]

→ Exemplo do java:

```
int numberOfMuppets = 10;  
numberOfMuppets = "ten";  
// java: incompatible types: java.lang.String cannot be  
// converted to int  
int numberOfCookies = 20;  
numberOfCookies = "ten" + numberOfCookies;  
// java: incompatible types: java.lang.String cannot be  
// converted to int
```

# Tipagem fraca



## Tipagem fraca [2]

- Regras de tipagem mais leves
- Geralmente associadas a linguagens interpretadas\*
- Existem casos mistos:
- C é considerado “misto”, por permitir conversão de tipos por ponteiros
- Python é dinamicamente tipada\*, mas com tipagem forte

# Tipagem fraca [1]

## → Desvantagens

- ◆ Não previne erros em tempo de execução
- ◆ Atraso em tempo de execução para definição de tipos
- ◆ Código é menos otimizado para o compilador
- ◆ Mais bugs, o que, no geral, atrapalha o desenvolvimento

# Tipagem fraca [1]

## → Vantagens

- ◆ Flexibilidade para programadores
- ◆ Variável pode ser atribuída a qualquer tipo

## Tipagem fraca [2]

→ Exemplo JavaScript

```
4 + '7'; // '47'
```

```
4 * '7'; // 28
```

```
2 + true; // 3
```

```
false - 3; // -3
```



# Tipagem estática

## Tipagem estática [2]

- O tipo é “preso” à variável
- Tipos são checados em tempo de compilação
- Uma vez que a variável recebe um tipo, não pode ser mudada, só atribuir novos valores (do mesmo tipo)

## Tipagem estática [2]

### → Vantagens

- ◆ Encontra e mostra erros de tipagem em compilação
- ◆ Ajuda a ter confiança no código
- ◆ Facilita testagem e robustez

### → Desvantagens

- ◆ Código precisa ser trabalhado com mais cuidado

## Tipagem estática [2]

→ Exemplo em Java:

```
String s = "hello";  
System.out.println(s) // "hello"  
s = "world";  
System.out.println(s) // "world"  
  
s = 5;  
System.out.println(s) // ERRO
```



# Tipagem dinâmica

## Tipagem dinâmica [2]

- Tipo da variável pode mudar de acordo com atribuição
- Permite maior flexibilidade de código
- Mas só é possível identificar os erros rodando

## Tipagem dinâmica [2]

### → Vantagens

- ◆ Útil para prototipação e iteração rápida de ideias
- ◆ Concentra-se mais na ideia que nos detalhes de implementação

### → Desvantagens

- ◆ Mais lentas que as compiladas (quando interpretadas)
- ◆ Erros passam a tempo de execução, dificultando testes e exigindo estratégias mais complexas

## Tipagem dinâmica [2]

→ Exemplo em Python

```
x = 1
```

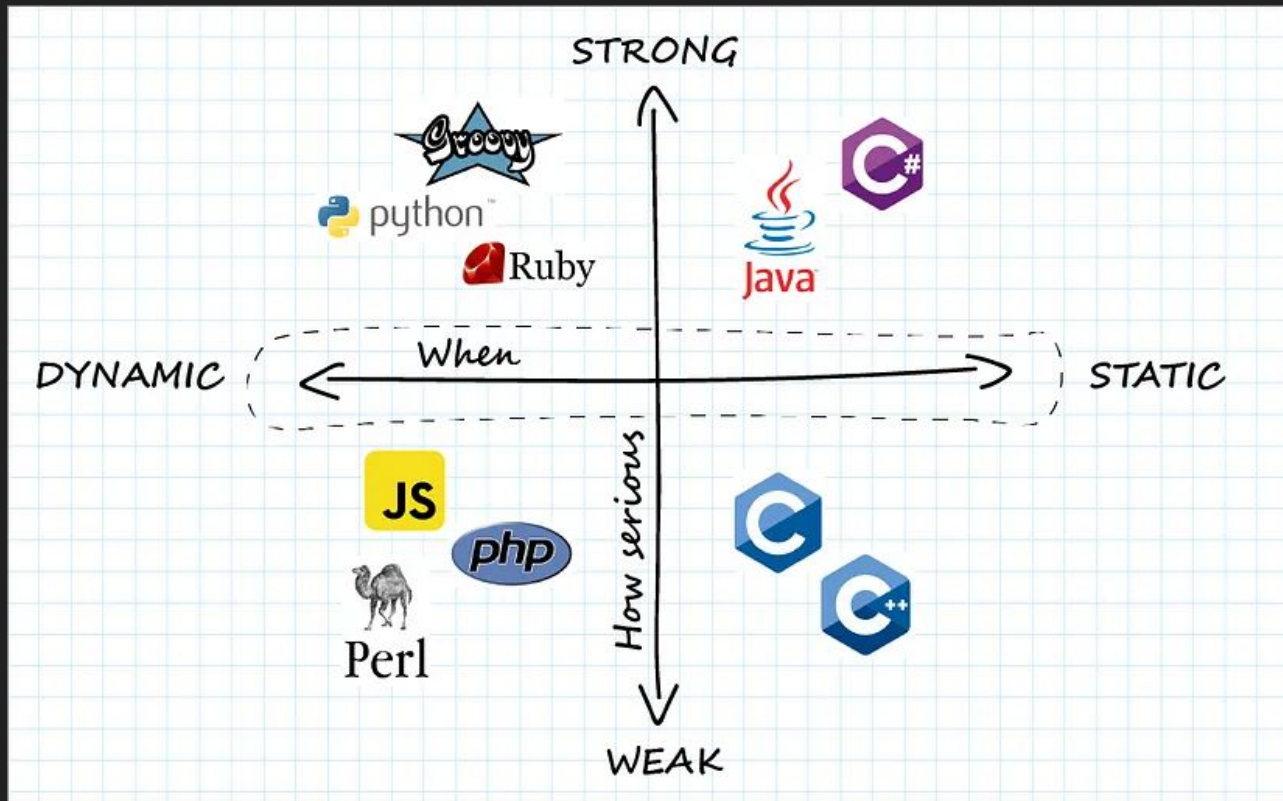
```
print(type(x)) # <class 'int'>
```

```
x = "hello"
```

```
print(type(x)) # <class 'str'>
```

```
x = 3.14
```

```
print(type(x)) # <class 'float'>
```



Comparação de linguagens por tipagem. Fonte [1]

# Referências

# Referências

1. <https://levelup.gitconnected.com/type-checking-explored-677f8673fbda>
2. <https://dev.to/leolas95/static-and-dynamic-typing-strong-and-weak-typing-5b0m>