Lógica e Linguagens de Programação

2o. ADS

Semana 1

Lógica e Algoritmos

Lógica

É o estudo dos princípios e regras do raciocínio. Ela é baseada em um **conjunto de regras bem definidas** que nos permitem **analisar, avaliar e raciocinar** sobre informações de forma **consistente e coerente**

Lógica de Programação

Permite **analisar e decompor problemas complexos em etapas lógicas e sequenciais** através das **instruções de uma linguagem de programação.**

Conceito de Algoritmo

Sequência de instruções bem definidas e organizadas que descrevem passo a passo como resolver um problema ou realizar uma tarefa específica.

Características

- Define ordem das operações
- Condições a serem verificadas
- Ações a serem tomadas

Algoritmo Eficiente

- Clareza das instruções
- Otimização de Recursos
- Resultado Correto

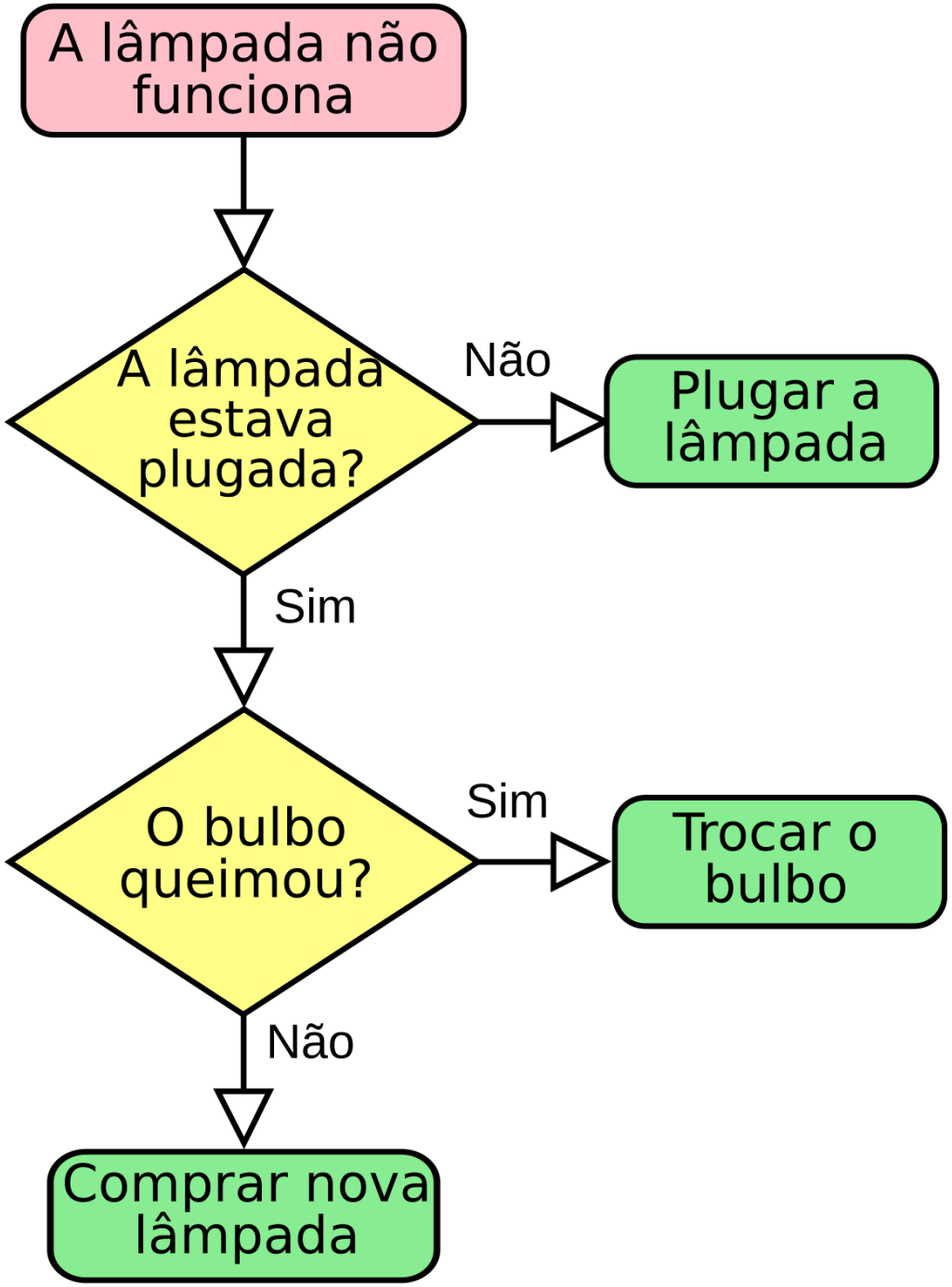
Estrutura básica de um algoritmo

- Deve sempre começar com "INÍCIO" e finalizar com "FIM"

```
ALGORITMO: Ir para a escola
INÍCIO
    1. Acordar
    2. Abrir o armário para escolher uma roupa
    3. Escolher uma roupa
    4. Ir ao banheiro para tomar banho
    5. Tomar banho e se secar
    6. Vestir a roupa escolhida
    7. Ir até a cozinha para tomar café
    8. Ir ao banheiro para escovar os dentes
    9. Pegar os materias para levar para a escola
    10. Sair de casa
    11. Pegar uma condução
    12. Descer na escola
FIM
```

Características Principais de um Algoritmo

- **Entrada:** Zero ou mais valores fornecidos antes do início.
- **Saída:** Um ou mais valores produzidos ao final.
- **Finitude:** Deve sempre terminar após um número finito de passos.
- **Definitividade:** Cada passo deve ser precisamente definido.
- **Efetividade:** Cada passo deve ser básico e realizável em tempo finito.



Fluxograma

Forma visual de se construir um algoritmo através de um diagrama com formas geométricas

Lógica e Linguagens de Programação

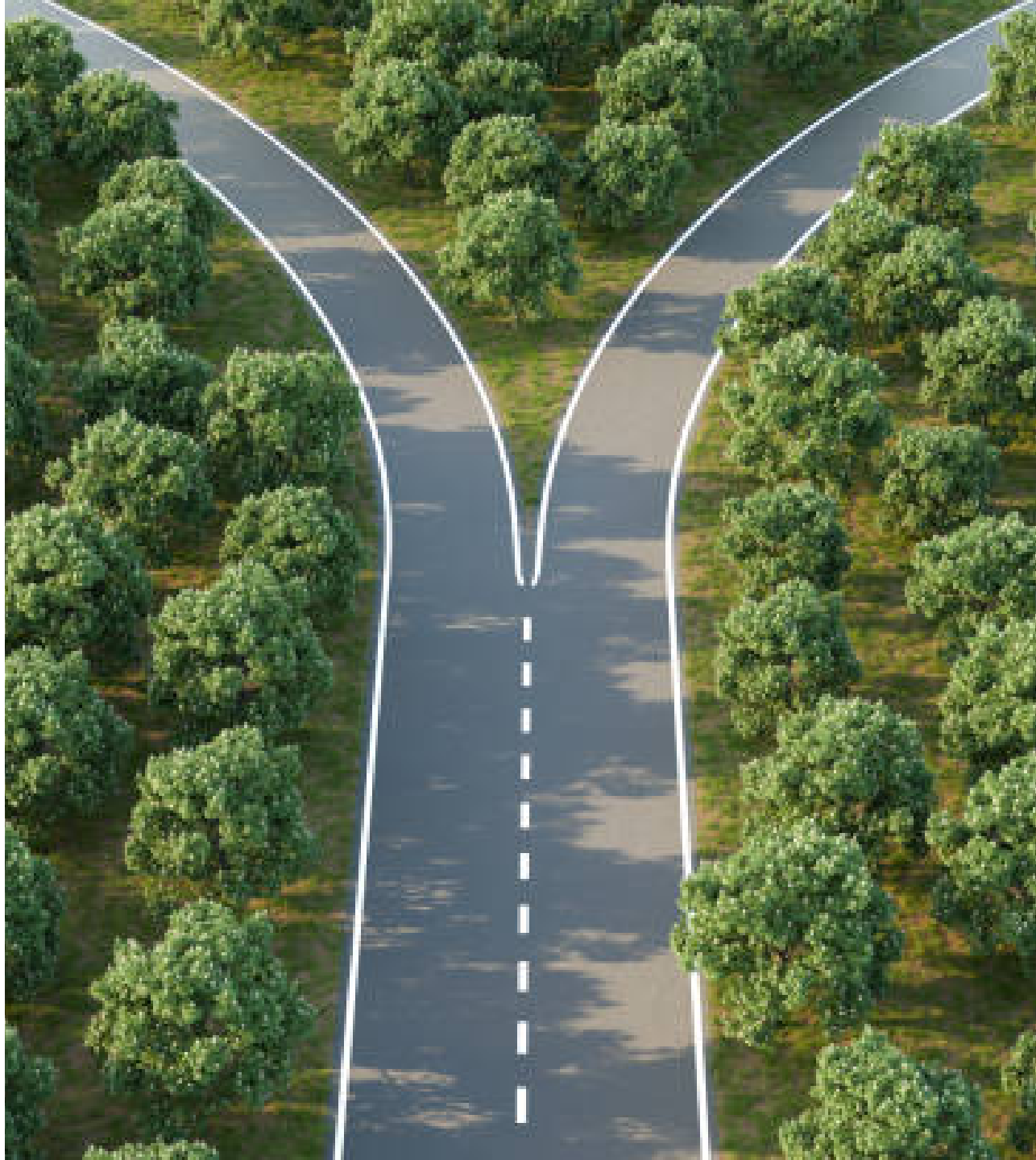
2o. ADS

Semana 2 e Semana 3

Estruturas de Decisão Simples e Composta

Estruturas de Decisão

Avaliam condições e direcionam a execução do programa para diferentes caminhos, permitindo adaptação dinâmica e automação de tarefas com tomada de decisão



Estruturas de Decisão Simples

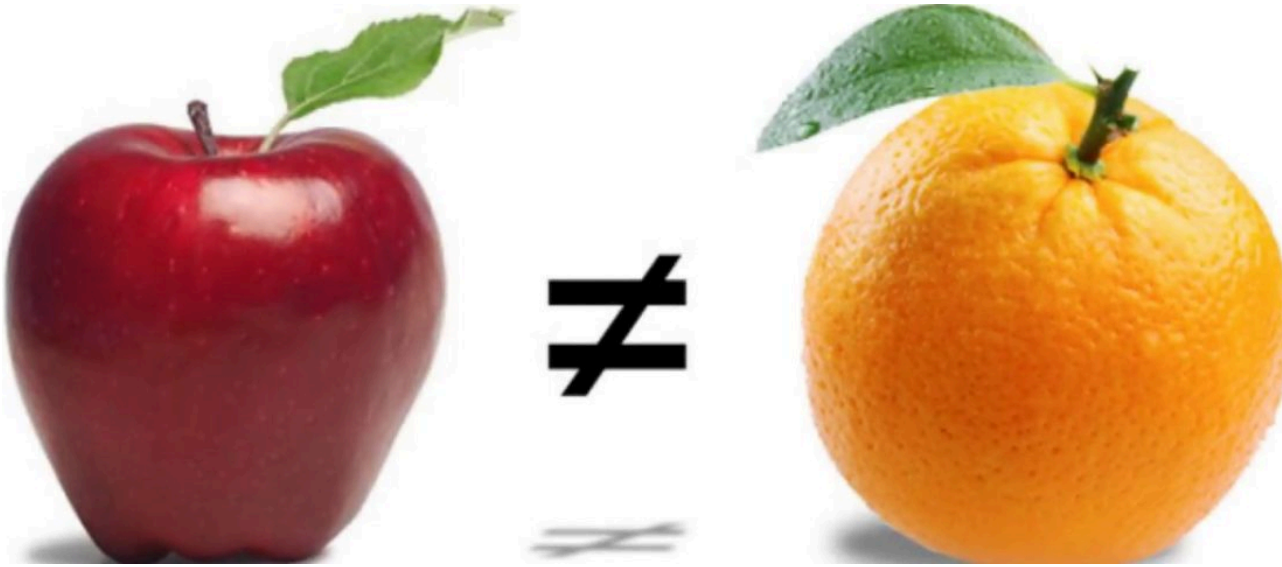
Quando existe apenas **uma condição para ser avaliada**

Comparadores

São eles que permitem comparar valores e expressões permitindo a tomada de decisões

Tipos de comparadores

- igualdade (**==**)
- desigualdade (**!=**)
- maior que (**>**)
- menor que (**<**)
- maior ou igual a (**>=**)
- menor ou igual a (**<=**)



Estruturas de Decisão Simples - Exemplo

```
INICIO
    VAR idade
    LEIA(idade)
    SE idade < 13 ENTÃO
        IMPRIMA("Criança") --> SE VERDADEIRO EXECUTA ESTE BLOCO
    SE NÃO
        IMPRIMA("Não é criança") --> SE FALSO ESTE
FIM
```

Python

```python
idade = int(input("QUAL A SUA IDADE?"))
if idade < 13:
    print("Criança")
else:
    print("Não é criança")
```

Estruturas de Decisão Composta

Quando existem **duas ou mais condições para serem avaliadas**

Operadores Lógicos

Permitem a combinação e avaliação de mais de uma condição em uma mesma expressão

AND / && (E): Combina duas condições e retorna VERDADEIRO somente se ambos os valores forem verdadeiros.

OR / || (OU): Basta uma das opções ser verdadeira para retornar VERDADEIRO.

NOT / ! (NÃO): Inverte o valor de uma condição. Se a entrada é VERDADEIRA, a saída é FALSA, e vice-versa.

Operadores Lógicos - Exemplo E/OU

Lâmpada 1	Lâmpada 2	Lamp. 1 E Lamp. 2	Lamp. 1 OU Lamp. 2
 VERDADEIRO	 VERDADEIRO	 VERDADEIRO	 VERDADEIRO
 VERDADEIRO	 FALSO	 FALSO	 VERDADEIRO
 FALSO	 VERDADEIRO	 FALSO	 VERDADEIRO
 FALSO	 FALSO	 FALSO	 FALSO

Operadores Lógicos - Exemplo NÃO

Lâmpada 1	NÃO Lâmpada 1
 VERDADEIRO	 FALSO
 FALSO	 VERDADEIRO

Lâmpada 2	NÃO Lâmpada 2
 FALSO	 VERDADEIRO
 VERDADEIRO	 FALSO

Estruturas de Decisão Composta - Exemplo com operadores lógicos

```
INICIO
    VAR idade
    LEIA(idade)
    SE idade < 13 OU idade < 18 ENTÃO
        IMPRIMA("Criança/Adolescente") --> SE VERDADEIRO EXECUTA ESTE BLOCO
    SE NÃO
        IMPRIMA("Adulto") --> SE FALSO EXECUTA ESTE
FIM
```

Python

```python
idade = int(input("QUAL A SUA IDADE?"))
if idade < 13 or idade < 18:
    print("Criança/Adolescente")
else:
    print("Adulto")
```

Estruturas de Decisão Composta - Exemplo usando ELSE IF (ELIF)

```
INICIO
    VAR idade
    LEIA(idade)
    SE idade < 13 ENTÃO
        IMPRIMA("Criança") --> SE VERDADEIRO EXECUTA ESTE BLOCO
    SE NÃO SE idade < 18 ENTÃO
        IMPRIMA("Adolescente") --> SE VERDADEIRO EXECUTA ESTE BLOCO
    SE NÃO
        IMPRIMA("Adulto") --> SE AMBOS ACIMA FALSOS, EXECUTA ESTE BLOCO
FIM
```

Python

```python
idade = int(input("QUAL A SUA IDADE?"))
if idade < 13:
    print("Criança")
elif idade < 18:
    print("Adolescente")
else:
    print("Adulto")
```

Lógica e Linguagens de Programação

2o. ADS

Semana 4

Estruturas de Seleção

Estruturas de Seleção

Estruturas de seleção **também são estruturas que permitem que através de condições fazer tomadas de decisões** em um programa, no funcionamento, **elas são iguais a uma estrutura de decisão, porém geralmente só permitem a verificação de uma condição específica e geralmente também deixam o código melhor legível**

Em muitas linguagens de programação ela é representada pelas palavras-chave **switch..case**

No Python antes da versão 3.10, eram representadas usando **if..elif..else**, porém a partir dessa versão foi criada a estrutura **match..case**

Estrutura de Seleção - Exemplo

Antes do Python 3.10

```
op = int(input("ESCOLHA A OPÇÃO DESEJADA:\n1 - BIG MAC\n2 - QUARTEIRÃO COM QUEIJO\n3 - MCNIFICO BACON"))
if op == 1:
    print("VOCÊ ESCOLHEU BIG MAC!")
elif op == 2:
    print("VOCÊ ESCOLHEU QUARTEIRÃO COM QUEIJO!")
elif op == 3:
    print("VOCÊ ESCOLHEU MCNIFICO BACON!")
else:
    print("VOCÊ NÃO ESCOLHEU NADA!")
```

A partir do Python 3.10

```
op = int(input("ESCOLHA A OPÇÃO DESEJADA:\n1 - BIG MAC\n2 - QUARTEIRÃO COM QUEIJO\n3 - MCNIFICO BACON"))
match op:
    case 1:
        print("VOCÊ ESCOLHEU BIG MAC!")
    case 2:
        print("VOCÊ ESCOLHEU QUARTEIRÃO COM QUEIJO!")
    case 3:
        print("VOCÊ ESCOLHEU MCNIFICO BACON!")
    case _:
        print("VOCÊ NÃO ESCOLHEU NADA!")
```

Lógica e Linguagens de Programação

2o. ADS

Semana 5

Estruturas de Repetição

Estruturas de repetição

Estruturas de repetição ou estruturas de **laço (ou loop)** permitem que um programa execute o mesmo trecho de código por um número fixo de vezes ou quando uma determinada condição de parada é atingida

Laços fixos e condicionais

- **Laços fixos:** São usados **quando se sabe previamente quantas vezes** o código deve ser executado. A repetição é baseada em um contador ou em uma coleção de elementos. Geralmente representando pela palavra-chave **for** (PARA)
- **Laços condicionais:** São utilizados **quando o número de repetições depende de condições que podem não ser conhecidas até que o código seja executado**. Eles são muito úteis para situações como leitura de dados até que um determinado valor seja encontrado, ou processamento contínuo até que uma condição externa seja satisfeita. Geralmente representando pelas palavras-chaves **do..while** (FAÇA..ENQUANTO), porém em Python só usamos a palavra-chave **while**

Estruturas de repetição

Laço Fixo

```
for numero in range(20):
    frase = "O NÚMERO É " + numero
    print(frase)
```

A função **range** é usada para criar um **contador fixo que vai de 1 a 20**

Laço Condicional

```
senha = input("Digite uma senha: ")
while senha != "1234":
    print("Senha incorreta!")
    senha = input("Digite uma senha: ")
```

Lógica e Linguagens de Programação

2o. ADS

Semana 6

Entrada, Processamento e Saída

Entrada

A "entrada" refere-se a todas as informações que um programa recebe para ser processado. Os tipos de dados que podem servir como entrada incluem texto digitado, números fornecidos por sensores, informações lidas de arquivos, cliques de mouse, entre outros.

No Python, por exemplo, quando queremos solicitar a entrada de um texto digitado usamos a função `input`

Tipos de dados de Entrada

- **Numéricos:** Inteiros (para contagem e indexação) e Pontos flutuantes (para precisão fracionária).
- **Alfanuméricos:** Caracteres (letras e símbolos) e Strings (sequências de caracteres para texto).
- **Booleanos:** Verdadeiro (true) ou Falso (false) para condições e estados.
- **Data e Hora:** Representações de datas e horários.
- **Binários e Dados de Arquivos:** Sequências de bits e conjuntos de dados tratados como unidades.

Processamento

O "processamento" é a etapa onde os dados de entrada são manipulados e transformados através de algoritmos e estruturas de controle para se obter um resultado. Durante o processamento, podem ocorrer operações matemáticas, manipulação de texto, comparações de valores e tomadas de decisão.

Saída

A "saída" é o resultado final do processamento realizado por um programa, a informação gerada e exibida. A saída pode ser apresentada como texto no console, imagens, sons, ações em dispositivos físicos ou pode ser armazenada em arquivos.

No Python, por exemplo, podemos usar a função `print` para apresentar um resultado (saída) para o usuário

Fluxogramas

Como já vimos anteriormente, geralmente são utilizados para mostrar de uma forma visual todo fluxo de entrada, processamento e saídas de um algoritmo/programa

Símbolos do Fluxograma

	**Terminal** – símbolo utilizado como ponto para indicar o início e/ou o fim do fluxo.
	**Seta do fluxo de dados** – indica o sentido do fluxo.
	**Processamento** – símbolo ou blocos que se utiliza para indicar cálculos (algoritmos) a efetuar, atribuições de valores ou qualquer manipulação de dados.
	**Entreda de dados** – utilizado para ler os dados necessários ao programa.
	**Saída de dados** – utiliza-se este símbolo quando se quer exibir os dados na tela.
	**Decisão** – indica a decisão que deve ser tomada, indicando a possibilidade de desvios para diversos outros pontos do fluxo, dependendo do resultado de comparação e de acordo com a situação das variaveis.