

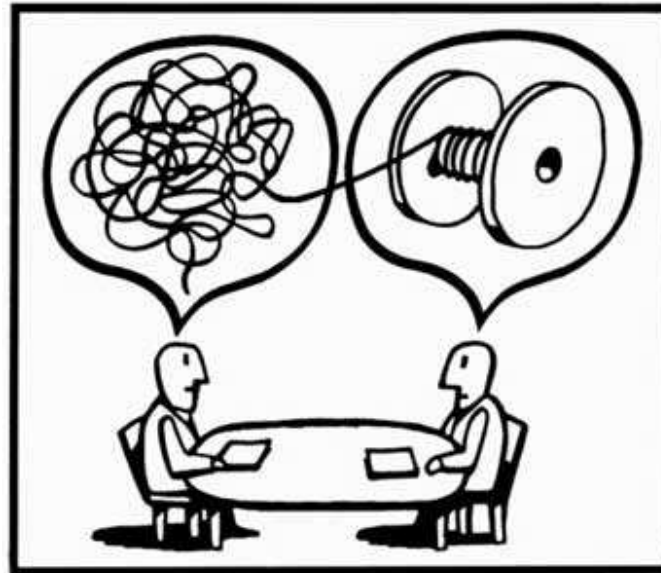
# Introdução à Programação



# Processo de resolução de problemas (Princípios de Pólya)

- Definição dos requisitos do problema (fazer o programa certo)
  - Entradas
  - Cálculos
  - Casos especiais
  - Saídas
- Desenvolvimento do algoritmo da solução (fazer certo o programa)
  - Português estruturado
  - Pseudocódigo
  - Fluxograma
- Codificação do programa
  - Python
- Teste do programa
  - Instrução com erro de grafia (defeito na codificação)
  - Resultado errado (defeito no algoritmo)

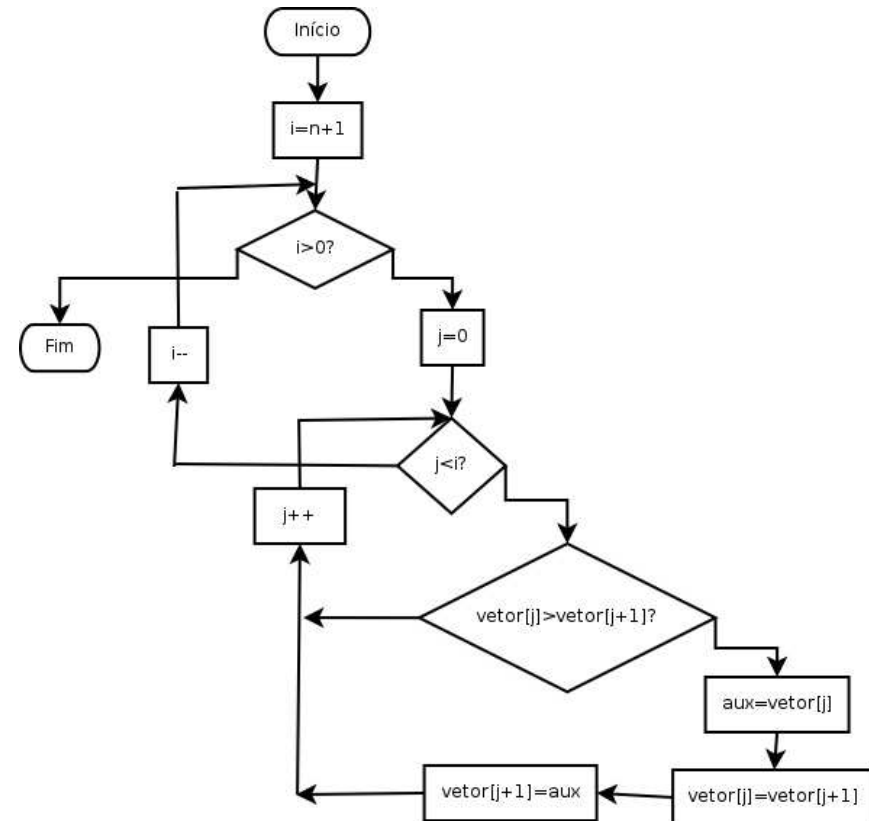
# Passo 1: Requisitos



Qual é o problema a ser resolvido?

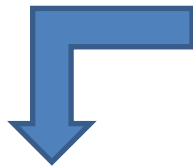
# Passo 2: Algoritmo

- Conjunto de **ações** para a **resolução** de um problema em um **número finito** de passos
- Parte mais complexa da programação
- Somente iniciar a programação quando
  - Souber qual problema deve ser resolvido
  - Souber como resolver o problema



# Passo 2: Algoritmo

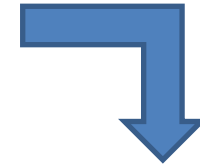
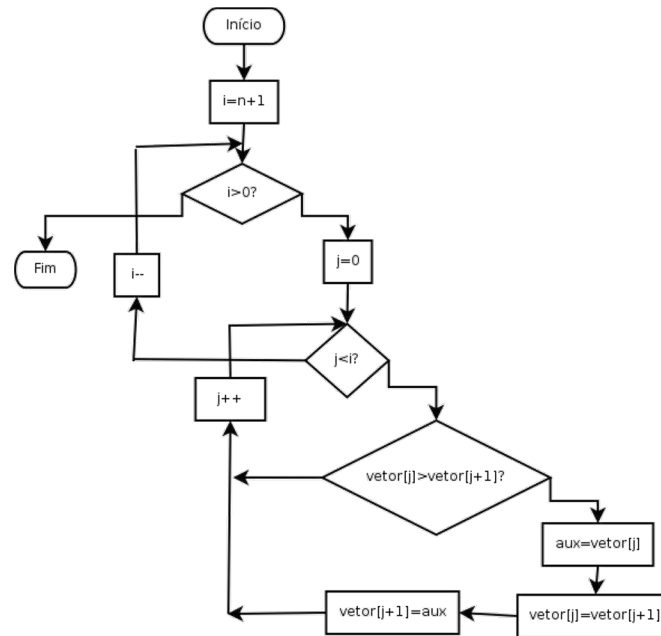
- Independente de linguagem de programação
- Pode ser implementado em diferentes linguagens



C++

```
#include <algorithm>
using namespace std;

void bubblesort(int a[], int n)
{
    for(int j=0; j<n; j++){
        for(int i=0; i<n-1; i++){
            if(a[i+1] < a[i])
                swap(a[i+1], a[i]);
        }
    }
}
```

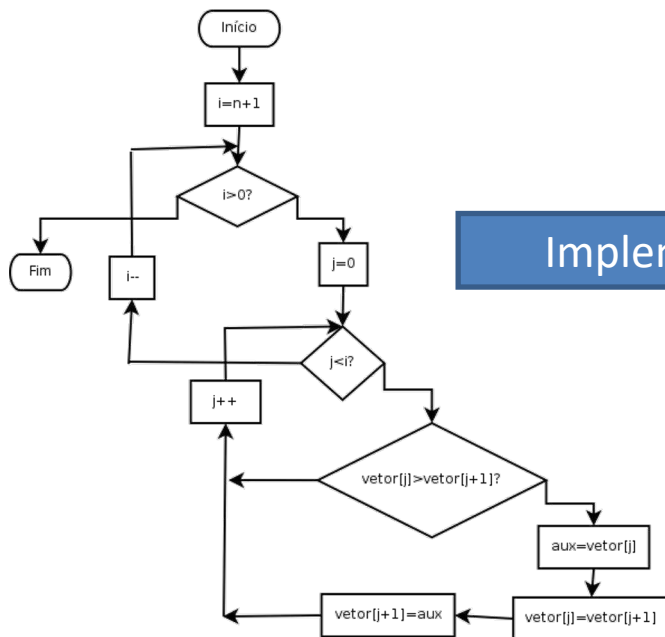


Matlab

```
for(i = 1:n-1)
    for(j = 1:n-i)
        if(x(j) > x(j + 1))
            aux = x(j);
            x(j) = x(j + 1);
            x(j + 1) = aux;
        end
    end
end
```

# Passo 3: Codificação

- A partir do algoritmo, traduzir (implementar) para a linguagem desejada
  - No nosso caso, Python



Implementação

Python

```

def bubble (vetor):
    houvetroca = True
    while (houvetroca):
        houvetroca = False
        for i in range(len(vetor) - 1):
            if (vetor[i] > vetor[i+1]):
                aux = vetor[i+1]
                vetor[i+1] = vetor[i]
                vetor[i] = aux
                houvetroca = True
    return v
  
```

# Por que não executar diretamente o algoritmo no computador?

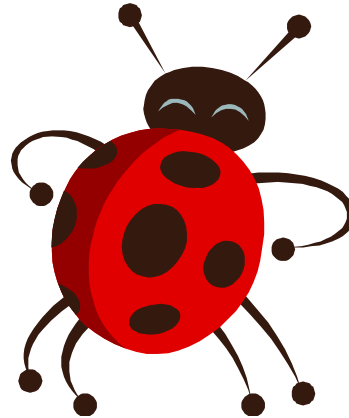
- Algoritmo é escrito em linguagem natural
- Linguagem natural é  **muito complexa e pouco precisa**
- É necessário usar uma linguagem mais simples e precisa, que o computador compreenda

“Calcule cinco mais cinco vezes dez”



# Passo 4: Teste

- O trabalho não termina com o código
- Todo código pode ter defeito (*bug*)
- Testar o código é fundamental!





# Tipos de erros

- Erro de sintaxe
  - Falha na tradução do algoritmo para Python
  - O compilador/interpretador vai detectar e dar dicas
  - Mais fáceis de corrigir
- Erro de lógica
  - Resultados diferentes do esperado
  - Erro de projeto do algoritmo
  - Mais difíceis de detectar e corrigir

# Exercício

- Escreva um algoritmo que consiga colocar em ordem as cartas de um naipe do baralho



# Algoritmos clássicos:

## *Insertion Sort*

Pegue a pilha de cartas desordenada

Enquanto existir carta na mão faça

    Pegue a primeira carta da mão

    Se não tem carta sobre a mesa então

        Coloque-a sobre a mesa

    Caso contrário

        Coloque-a na posição correta da pilha da mesa

# Algoritmos clássicos:

## *Selection Sort*

Pegue a pilha de cartas desordenada

Enquanto existir carta na mão faça

    Pegue a maior carta da mão

    Se não tem carta sobre a mesa então

        Coloque-a sobre a mesa

    Caso contrário

        Coloque-a no topo da pilha da mesa

# Algoritmos clássicos: *Bubble Sort*

Pegue a pilha de cartas desordenada

Enquanto as cartas não estiverem ordenadas faça

Para cada carta do baralho faça

Se a carta seguinte for menor que a carta atual

Inverta a posição destas cartas

# Algoritmos clássicos:

## *Bogo Sort*

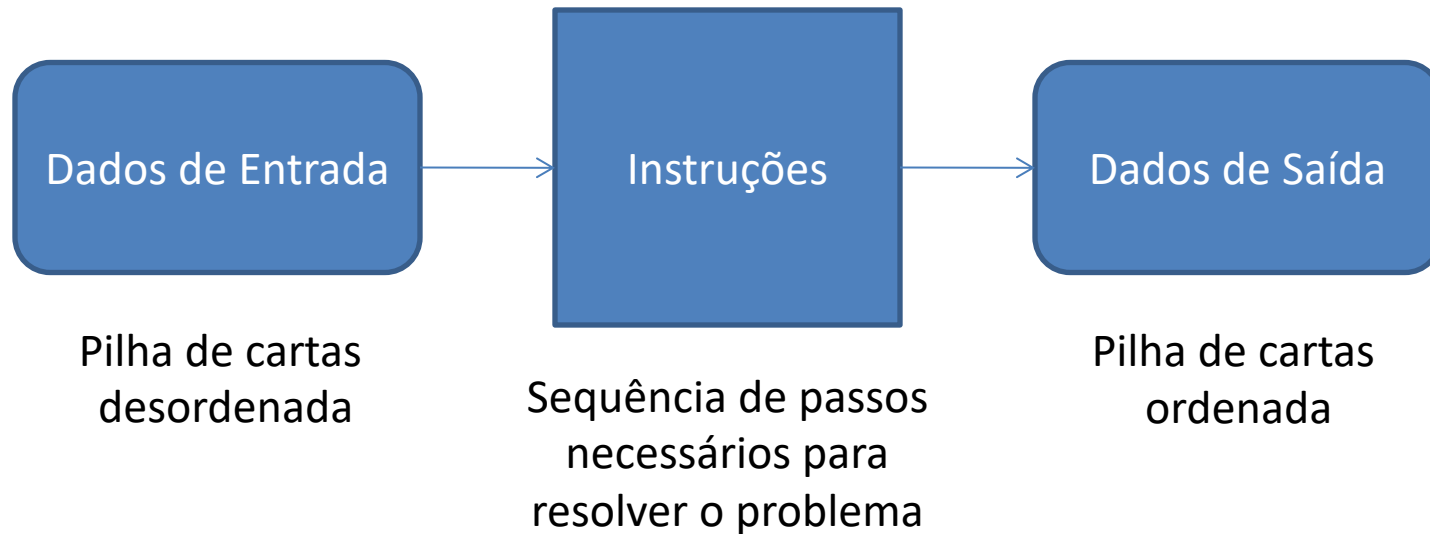
Pegue a pilha de cartas desordenada

Enquanto as cartas não estiverem ordenadas faça

Arremesse as cartas para cima

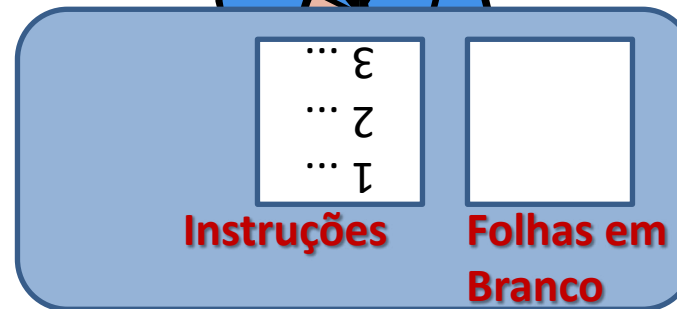
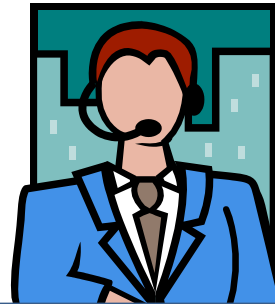
Recolha as cartas do chão de forma aleatória

# E se tivermos que pedir para o computador resolver?



# Analogia: Secretária

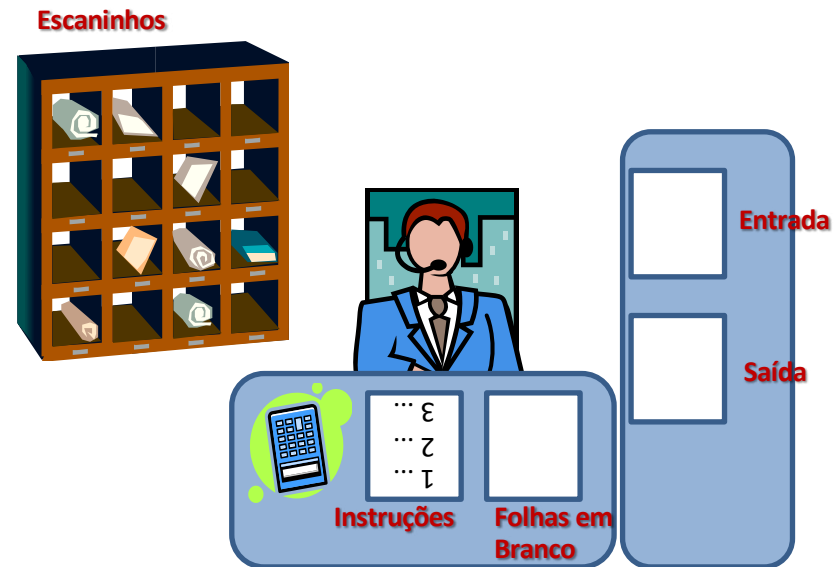
## Escaninhos





# Analogia: Secretária

- Secretária conhece um conjunto pequeno de instruções
- Ela segue as instruções ao pé da letra
- Cada escaninho tem uma etiqueta com um “rótulo”
- No fim do dia, o *boy* passa e limpa os escaninhos



# Analogia: Secretária

- O que a secretária sabe fazer (instruções)
  - Ler um valor de um escaninho ou da caixa de entrada
  - Escrever um valor em um escaninho ou na caixa de saída
  - Calcular (somar, subtrair, multiplicar, dividir)
  - Avaliar uma expressão, gerando como resultado **verdadeiro** ou **falso**

# Algoritmo para somar dois números

Leia um valor da caixa de entrada

Escreva esse valor no escaninho A

Leia um valor da caixa de entrada

Escreva esse valor no escaninho B

Some o valor do escaninho A com o valor do escaninho B

Escreva o resultado no escaninho SOMA

Leia o valor do escaninho SOMA

Escreva na caixa de saída

# Instrução “Avalie”

- Avalia uma expressão e indica se ela é verdadeira ou falsa
  - Avalie  $2 = 3 \rightarrow$  falso
  - Avalie  $10 > 5 \rightarrow$  verdadeiro
- Conector lógico “e”: todos os itens avaliados devem ser verdadeiros para a expressão ser verdadeira
  - Avalie  $10 > 5$  e  $2 = 3 \rightarrow$  falso
- Conector lógico “ou”: basta que um dos itens seja verdadeiro para que a expressão seja verdadeira
  - Avalie  $10 > 5$  ou  $2 = 3 \rightarrow$  verdadeiro

# Algoritmo para indicar se um número é maior que outro

Leia um valor da caixa de entrada

Escreva esse valor no escaninho A

Leia um valor da caixa de entrada

Escreva esse valor no escaninho B

Avalie  $A > B$

Escreva o resultado no escaninho R

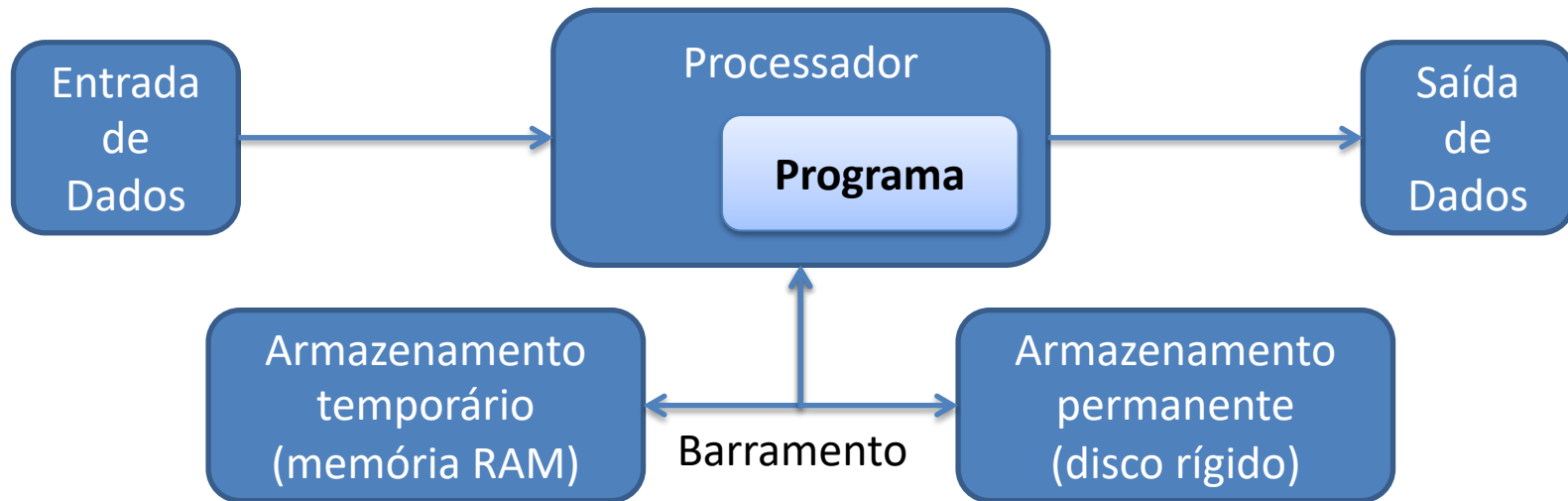
Leia o valor do escaninho R

Escreva o valor do escaninho R na caixa de saída

# Secretária x Computador

- Secretária é a CPU do computador (quem executa as instruções)
- Instruções são os programas
- Escaninhos são as posições na memória RAM do computador
- Caixa de Entrada é o teclado
- Caixa de Saída é o monitor
  
- O *boy* no fim do dia esvazia o escaninho: Memória RAM do computador é volátil (apaga se o computador for desligado)

# Arquitetura de um computador



Entrada	Saída	Armazenamento
Teclado	Vídeo	Memória
Mouse	Impressora	Discos rígidos
Scanner	Auto-Falante	CD/DVD
Webcam		Pen drive

# Pseudocódigo

- Forma genérica, mas sucinta, para escrever um algoritmo
- Fácil para um humano entender
- Fácil de ser codificada
- Voltando aos exemplos anteriores

```
Leia A  
Leia B  
SOMA ← A + B  
Escreva SOMA
```

```
Leia A  
Leia B  
R ← A > B  
Escreva R
```



# Exercício

- Em relação ao pseudocódigo a seguir

```
Leia Valor
```

```
Leia Quantidade
```

```
Total ← Valor * Quantidade
```

```
Escreva Total
```

- Quais são os dados de entrada e saída?
- Quais linhas são somente de processamento?

# Exercício

- Qual é a funcionalidade desse algoritmo?  
Execute para os valores 25 e 7.

```
Leia A
Leia B
C ← 0
Enquanto A >= B faça {
    A ← A - B
    C ← C + 1
}
Escreva C
Escreva A
```

# Exercício

- Escreva um algoritmo em pseudocódigo para
  - a) Somar três números
  - b) Calcular a média de um aluno numa disciplina, sendo  
$$\text{Média} = (\text{Provas} + 3 \times \text{Trabalho} + \text{Participação}) / 10$$
$$\text{Provas} = 3 \times \text{Prova1} + 3 \times \text{Prova2}$$
  - c) Calcular o peso ideal de uma pessoa, assumindo  
Homem:  $\text{Peso} = (72,7 * \text{Altura}) - 58$   
Mulher:  $\text{Peso} = (62,1 * \text{Altura}) - 44,7$

# Exercício

- Escreva um algoritmo para separar o líquido de três garrafas com formatos diferentes em duas quantidades iguais, onde
  - A garrafa A está cheia e tem capacidade de 8 litros
  - A garrafa B está vazia e tem capacidade de 5 litros
  - A garrafa C está vazia e tem capacidade de 3 litros

# Exercício

- Escreva um algoritmo para descobrir a moeda falsa (mais leve) de um total de 5 moedas usando uma balança analítica
  - Dica: é possível resolver com somente duas pesagens
- Idem ao anterior, mas com um total de 27 moedas
  - Dica: é possível resolver com somente três pesagens



# Referências

- Material feito em conjunto com Aline Paes e Vanessa Braganholo
- Alguns exercícios extraídos do livro Furlan, M., Gomes, M., Soares, M., Concilio, R., 2005, “Algoritmos e Lógica de Programação”, Editora Thomson.

# Introdução à Programação

