



**UNIVERSIDADE  
FEDERAL DO CARIRI**

# Introdução à Programação

Professora Dra. Luana Batista da Cruz  
[luana.batista@ufca.edu.br](mailto:luana.batista@ufca.edu.br)

# Roteiro

01 Introdução

02 Representação de algoritmos

01

# Introdução

Objetivo básico da computação  
Finalidade de um computador  
Conceitos preliminares  
Como construir um algoritmo

# Objetivo básico da computação

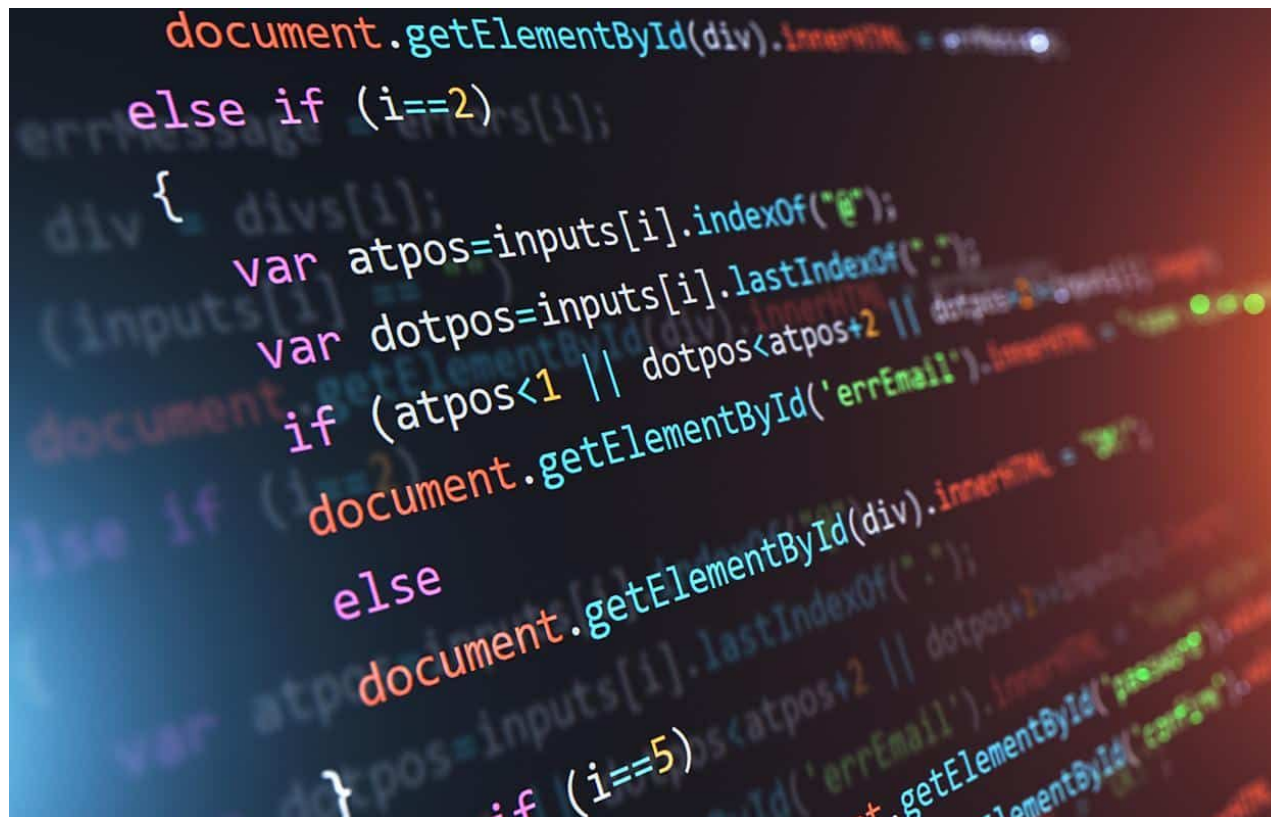
- Auxiliar os seres humanos em trabalhos repetitivos e manuais, diminuindo esforços e economizando tempo
- O computador é capaz de auxiliar em qualquer coisa que lhe seja solicitada, mas
  - Não tem iniciativa
  - Não é independente
  - Não é criativo nem inteligente
- É necessário que o computador receba suas instruções nos mínimos detalhes, para que tenha condições de realizar suas tarefas

# Finalidade de um computador

- O computador deve receber, manipular e armazenar dados (todas essas operações são realizadas por meio de programas)
- Quando construímos um software para realizar determinado processamento de dados, devemos escrever um programa ou vários programas interligados
- Para que o computador consiga ler o programa e entender o que fazer, este programa deve ser escrito em uma linguagem que o computador entenda  
→ LINGUAGEM DE PROGRAMAÇÃO

# Conceitos preliminares

- O que é algoritmo?



```
document.getElementById(div).innerHTML = errMessage;
else if (i==2)
{
var atpos=inputs[i].indexOf("@");
var dotpos=inputs[i].lastIndexOf(".");
if (atpos<1 || dotpos<atpos+2 || dotpos>inputs[i].lastIndexOf(".")-1)
document.getElementById('errEmail').innerHTML = "Email inválido";
else
document.getElementById(div).innerHTML = "OK";
}
if (i==5)
```

# Conceitos preliminares

- **Algoritmo**

- Conjunto das regras e procedimentos lógicos perfeitamente definidos que levam à solução de um problema em um número finito de etapas
- Em matemática e ciência da computação, um algoritmo é uma sequência finita de ações executáveis que visam obter uma solução para um determinado tipo de problema (Ziviani, 2011)

# Conceitos preliminares

- **Algoritmo**

- Não são operações exclusivas de um computador, pode ser aplicado a qualquer problema cuja solução possa ser decomposta em um grupo de instruções
- Exemplos de algoritmos
  - Instruções para se utilizar um aparelho eletrodoméstico
  - Uma receita para preparo de algum prato
  - Guia de preenchimento para declaração do imposto de renda
  - Fazer um sanduíche
  - Trocar uma lâmpada
  - Sacar dinheiro em um banco 24 horas



# Conceitos preliminares

- **Algoritmo x Programa**

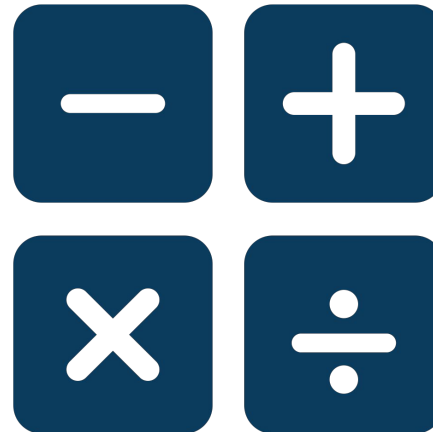
- Para a grande maioria dos problemas, é possível haver mais de um **algoritmo** para solucionar um determinado problema
- Um **programa** é um conjunto de milhares de instruções que indicam ao computador, passo a passo, o que ele tem que fazer
- Um **programa** nada mais é do que um **algoritmo computacional** descrito em uma **linguagem de programação**
  - Conjunto de instruções finito
  - Cada linguagem de programação tem sua sintaxe própria
  - Cada linguagem de programação tem um propósito de existir. Por exemplo: C (construir drivers), PHP (sistemas web)

# Conceitos preliminares

- **Características de um algoritmo**
  - Finitas
  - Bem definidas
  - Efetivas

# Conceitos preliminares

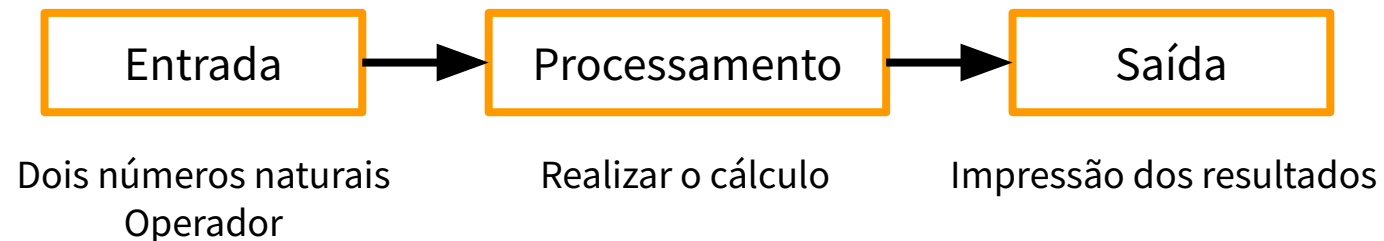
- **Características de um algoritmo**
  - **Finitas:** o algoritmo deve eventualmente resolver o problema
    - Ex: realizar cálculos com 4 operadores básicos



# Conceitos preliminares

- **Características de um algoritmo**

- **Bem definidas:** os passos devem ser definidos de modo a serem entendidos
  - Ex: realizar cálculos com 4 operadores básicos



# Conceitos preliminares

- **Características de um algoritmo**

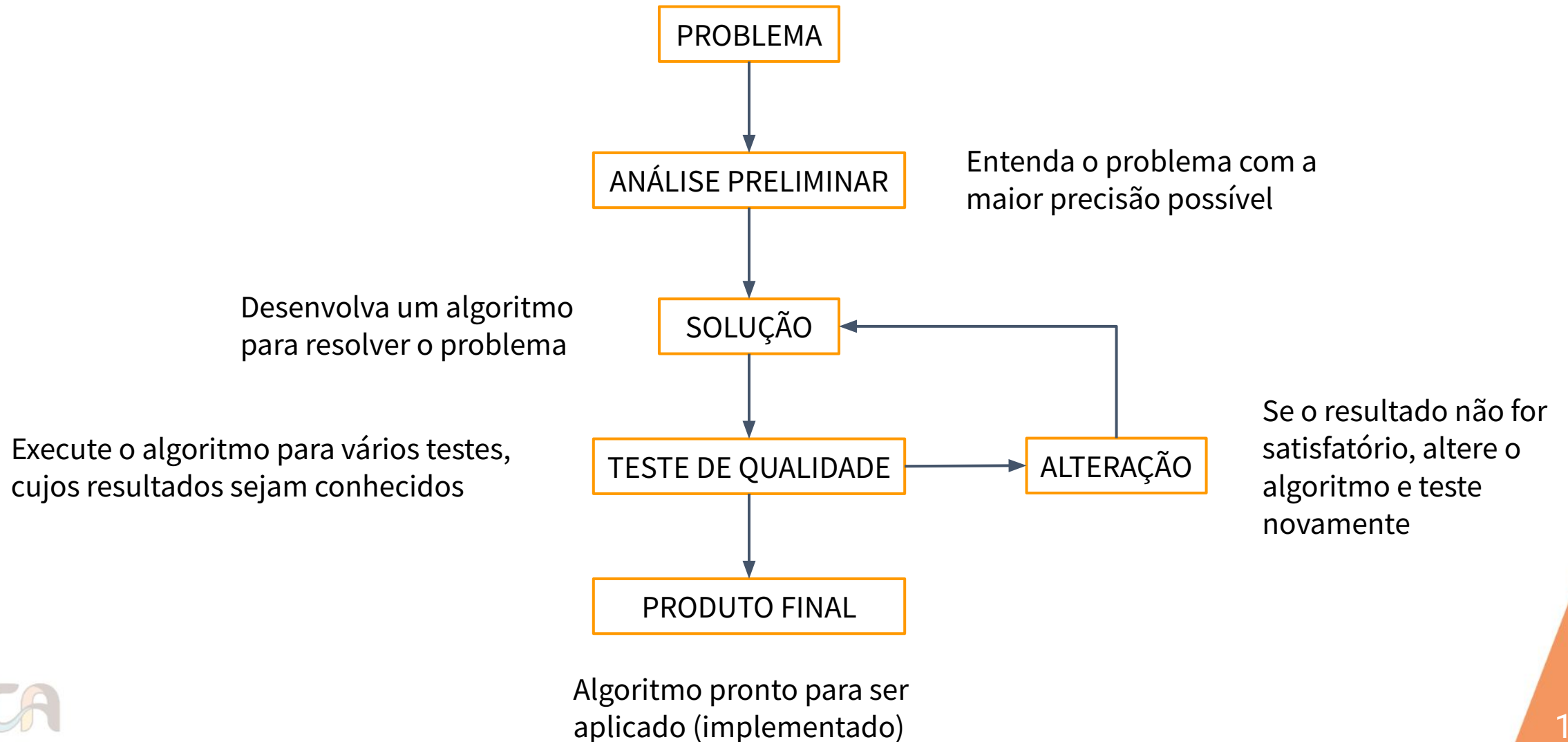
- **Efetivas:** deve sempre resolver o que tem para solucionar, antecipando falhas
  - Ex: realizar cálculos com 4 operadores básicos

|   |               |               |
|---|---------------|---------------|
| + | soma          | $3 + 2 = 5$   |
| - | subtração     | $3 - 2 = 1$   |
| * | multiplicação | $3 * 2 = 6$   |
| / | divisão       | $3 / 2 = 1.5$ |

# Conceitos preliminares

- **Características de um algoritmo**
  - **Em resumo**, todo algoritmo deve
    - Ter fim
    - Não dar margem à dupla interpretação (não ambíguo)
    - Ter capacidade de receber dado(s) de entrada do mundo exterior
    - Poder gerar informações de saída para o mundo externo ao do ambiente do algoritmo
    - Ser efetivo (todas as etapas especificadas no algoritmo devem ser alcançáveis em um tempo finito)

# Como construir um algoritmo



# Como construir um algoritmo

- **Treinando o cérebro**
  - O painel do meio no centro do alvo não tem número. O que o **X** representa?

|    |          |    |
|----|----------|----|
| 12 | 10       | 7  |
| 21 | <b>X</b> | 10 |
| 30 | 22       | 13 |



# Como construir um algoritmo

- **Treinando o cérebro**
  - O painel do meio no centro do alvo não tem número. O que o X representa?

|    |    |    |
|----|----|----|
| 12 | 10 | 7  |
| 21 | X  | 10 |
| 30 | 22 | 13 |

|    |    |    |
|----|----|----|
| 12 | 10 | 7  |
| 21 | 16 | 10 |
| 30 | 22 | 13 |

# Como construir um algoritmo

- **Treinando o cérebro**
  - Para a direita sou a espada que fere; para a esquerda sou a fera que é ferida. Que palavra sou eu? (ao contrário)

# Como construir um algoritmo

- **Treinando o cérebro**
  - Para a direita sou a espada que fere; para a esquerda sou a fera que é ferida. Que palavra sou eu? (ao contrário)

Lâmina/Animal

# Como construir um algoritmo

- **Treinando o cérebro**

- Que dois símbolos matemáticos (+, -, \*, /) você deve colocar nessa conta para obter 57?

$$76 \ ? \ 15 \ ? \ 20 = 57$$

# Como construir um algoritmo

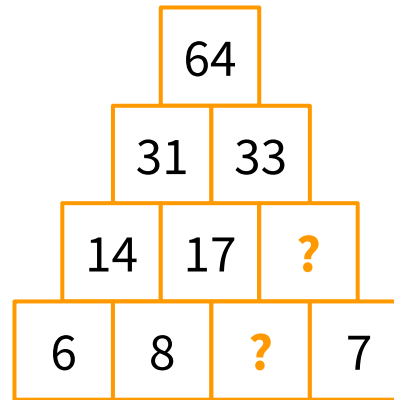
- **Treinando o cérebro**

- Que dois símbolos matemáticos (+, -, \*, /) você deve colocar nessa conta para obter 57?

$$\begin{array}{l} 76 \text{ ? } 15 \text{ ? } 20 = 57 \\ 76 * 15 / 20 = 57 \end{array}$$

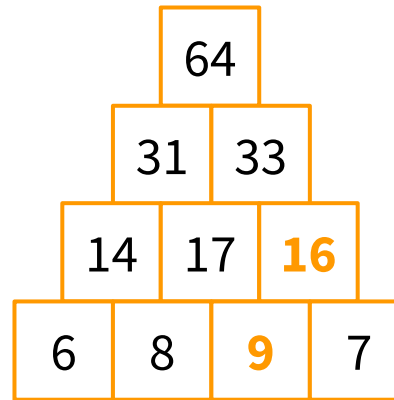
# Como construir um algoritmo

- **Treinando o cérebro**
  - Resolva a pirâmide numérica



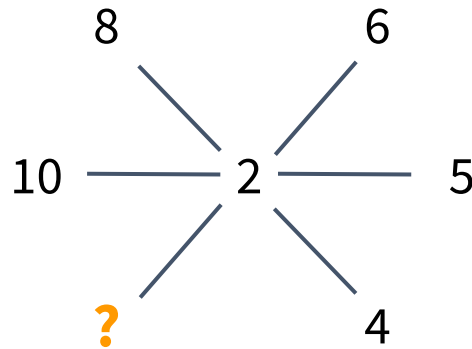
# Como construir um algoritmo

- **Treinando o cérebro**
  - Resolva a pirâmide numérica



# Como construir um algoritmo

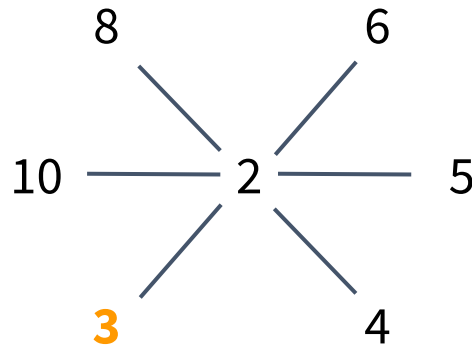
- **Treinando o cérebro**
  - Resolva esse quebra cabeça





# Como construir um algoritmo

- **Treinando o cérebro**
  - Resolva esse quebra cabeça



# Representação de algoritmos

Formas de representação de algoritmos

Descrição narrativa

Fluxograma

Linguagem algorítmica, pseudocódigo ou português estruturado (portugol)

# Representação de algoritmos

- Humanos: linguagem natural tradicional, linguagem de figuras
  - Problemas da linguagem natural
    - Mal entendimento: algumas vezes possui mais de um significado
    - Ex1.: “Caetano sentou na cadeira e quebrou o braço”
      - O braço que quebrou era de Caetano ou da cadeira?
    - Ex2.: “Ana pediu a Caio que pegasse sua carteira”
      - A carteira era de Ana ou Caio?
    - Ex3.: “A mãe avisou à filha que estava indo ao shopping”
      - Quem ia ao shopping, a mãe ou a filha?

# Representação de algoritmos

- Humanos: linguagem natural tradicional, linguagem de figuras
  - Problemas da linguagem natural
    - Mal entendimento: algumas vezes possui mais de um significado
    - Ex4.: “O garoto doente não conseguiu andar”
      - O garoto sempre teve uma doença e não conseguiu andar dessa vez, ou estava doente e por isso não conseguiu andar naquele momento?
    - Ex5.: “Maria olhou a gata correndo”
      - Quem estava correndo, Maria ou a gata?
- Problemas de comunicação surgem quando a linguagem usada para a representação de um algoritmo não é precisamente definida ou quando a informação não é dada com o detalhamento adequado

# Formas de representação de algoritmos

- Algoritmos podem ser representados em diversas formas
  - Descrição narrativa
  - Fluxograma
  - Linguagem algorítmica, pseudocódigo ou português estruturado

# Formas de representação de algoritmos

- **Descrição narrativa**

- Analisar o enunciado. Faz o uso de uma linguagem natural (ex: português) para descrever algoritmos

- Exemplo: receita de bolo

- Manteiga, ovos, trigo, etc
- Misture os ingredientes
- Despeje a mistura na forma de bolo
- Leve ao forno
- Espere 20 minutos
- Retire do forno
- Deixe esfriar



- **Vantagens**

- Linguagem natural é bastante conhecida

- **Desvantagens**

- Ambiguidades ou mal entendimento
  - Expressões podem possuir mais de um significado
- Pouca confiabilidade
- Extensão: normalmente, escreve-se muito para dizer pouca coisa

# Formas de representação de algoritmos

- **Descrição narrativa**
  - **Exercício 1:** elabore um algoritmo em um descrição narrativa que efetue a troca de uma lâmpada



# Formas de representação de algoritmos

- **Descrição narrativa**

- **Exercício 1:** elabore um algoritmo em um descrição narrativa que efetue a troca de uma lâmpada
  - Algoritmo
    1. Remova a lâmpada queimada
    2. Coloque uma nova lâmpada



# Formas de representação de algoritmos

- **Descrição narrativa**

- **Exercício 1:** elabore um algoritmo em um descrição narrativa que efetue a troca de uma lâmpada
  - Algoritmo
    1. Remova a lâmpada queimada
    2. Coloque uma nova lâmpada

Como melhorar o algoritmo?

# Formas de representação de algoritmos

- **Descrição narrativa**

- **Exercício 1:** elabore um algoritmo em um descrição narrativa que efetue a troca de uma lâmpada

- Algoritmo

1. Pegue uma escada
2. Posicione a escada embaixo da lâmpada
3. Busque uma lâmpada nova
4. Suba na escada
5. Retire a lâmpada velha
6. Coloque a lâmpada nova
7. Desça da escada
8. Guarde a escada

# Formas de representação de algoritmos

- **Descrição narrativa**

- **Exercício 1:** elabore um algoritmo em um descrição narrativa que efetue a troca de uma lâmpada

- Algoritmo

1. Pegue uma escada
2. Posicione a escada embaixo da lâmpada
3. Busque uma lâmpada nova
4. Suba na escada
5. Retire a lâmpada velha
6. Coloque a lâmpada nova
7. Desça da escada
8. Guarde a escada

E se a lâmpada não estiver queimada?  
Como seria o novo algoritmo?

# Formas de representação de algoritmos

- **Descrição narrativa**

- **Exercício 1**

- **Acionar o interruptor**

- **SE a lâmpada não acender, ENTÃO**

- 1. Pegue uma escada
        - 2. Posicione a escada embaixo de uma lâmpada
        - 3. Busque uma lâmpada nova
        - 4. Suba na escada
        - 5. Retire a lâmpada queimada
        - 6. Coloque a lâmpada nova
        - 7. Desça da escada
        - 8. Guarde a escada

# Formas de representação de algoritmos

- **Descrição narrativa**

- **Exercício 1**

- **Acionar o interruptor**

- **SE a lâmpada não acender, ENTÃO**

1. Pegue uma escada
2. Posicione a escada embaixo de uma lâmpada
3. Busque uma lâmpada nova
4. Suba na escada
5. Retire a lâmpada queimada
6. Coloque a lâmpada nova
7. Desça da escada
8. Guarde a escada

E se a lâmpada nova não funcionar?

# Formas de representação de algoritmos

- **Descrição narrativa**

- **Exercício 1**

- **Acionar o interruptor**

- **SE a lâmpada não acender, ENTÃO**

- 1. Pegue uma escada
        - 2. Posicione a escada embaixo de uma lâmpada
        - 3. Busque uma lâmpada nova
        - 4. Suba na escada
        - 5. Retire a lâmpada queimada
        - 6. Coloque a lâmpada nova

- **SE a lâmpada não acender, ENTÃO**

- 1. Retire a lâmpada queimada
        - 2. Coloque outra lâmpada nova

Até quando?

- **SE a lâmpada não acender, ENTÃO**

- 1. Retire a lâmpada queimada .
        - 2. Coloque outra lâmpada nova :

# Formas de representação de algoritmos

- **Descrição narrativa**

- **Exercício 1**

- **Acionar o interruptor**

- **SE a lâmpada não acender, ENTÃO**

- 1. Pegue uma escada
        - 2. Posicione a escada embaixo de uma lâmpada
        - 3. Busque uma lâmpada nova
        - 4. Suba na escada
        - 5. Retire a lâmpada queimada
        - 6. Coloque a lâmpada nova

- **ENQUANTO a lâmpada não acender, FAÇA**

- 1. Retire a lâmpada queimada
          - 2. Coloque outra lâmpada nova
          - 3. Desça da escada
          - 4. Guarde a escada

# Formas de representação de algoritmos

- **Descrição narrativa**

- **Exercício 1**

- **Acionar o interruptor**

- **SE a lâmpada não acender, ENTÃO**

1. Pegue uma escada
2. Posicione a escada embaixo de uma lâmpada
3. Busque uma lâmpada nova
4. Suba na escada
5. Retire a lâmpada queimada
6. Coloque a lâmpada nova

- **ENQUANTO a lâmpada não acender, FAÇA**

1. Retire a lâmpada queimada
2. Coloque outra lâmpada nova
3. Desça da escada
4. Guarde a escada

- Decisão para uma determinada condição
  - Estrutura condicional

- Evitar repetição de um determinado trecho de código
  - Estrutura de repetição



# Formas de representação de algoritmos

- **Descrição narrativa**

- **Exercício 1:** elabore um algoritmo em um descrição narrativa que efetue a troca de uma lâmpada (**outra solução**)

- Algoritmo

1. Coloque uma escada embaixo da lâmpada queimada
2. Escolha uma lâmpada nova de mesma potência/voltagem da queimada
3. Suba na escada até alcançar a lâmpada queimada
4. Gire a lâmpada queimada no sentido anti-horário até que ela se solte
5. Posicione a lâmpada nova no soquete
6. Gire a lâmpada no sentido horário, até que ela se firme
7. Desça da escada
8. Guarde a escada

# Formas de representação de algoritmos

- **Descrição narrativa**

- **Exercício 1:** elabore um algoritmo em um descrição narrativa que efetue a troca de uma lâmpada (**outra solução**)

- Algoritmo

1. Coloque uma escada embaixo da lâmpada queimada
2. Escolha uma lâmpada nova de mesma potência/voltagem da queimada
3. Suba na escada até alcançar a lâmpada queimada
4. Gire a lâmpada queimada no sentido anti-horário até que ela se solte
5. Posicione a lâmpada nova no soquete
6. Gire a lâmpada no sentido horário, até que ela se firme
7. Desça da escada
8. Guarde a escada

Onde podemos observar estruturas de repetição?

# Formas de representação de algoritmos

- **Descrição narrativa**

- **Exercício 1:** elabore um algoritmo em um descrição narrativa que efetue a troca de uma lâmpada (**outra solução**)

- Algoritmo

1. Coloque uma escada embaixo da lâmpada queimada
2. **Escolha uma lâmpada nova de mesma potência/voltagem da queimada**
3. **Suba na escada até alcançar a lâmpada queimada**
4. **Gire a lâmpada queimada no sentido anti-horário até que ela se solte**
5. Posicione a lâmpada nova no soquete
6. **Gire a lâmpada no sentido horário, até que ela se firme**
7. Desça da escada
8. Guarde a escada

Onde podemos observar estruturas de repetição?

# Formas de representação de algoritmos

- **Descrição narrativa**
  - **Exercício 1 (outra solução)**
    - **Acionar o interruptor**
      - **SE a lâmpada não acender, ENTÃO**
        1. Posicionar a escada embaixo de uma lâmpada
        - **ENQUANTO a potência/voltagem não for a mesma da queimada FAÇA**
          3. Descarte a lâmpada escolhida
          4. Escolha outra lâmpada
        - **ENQUANTO não alcançar a lâmpada queimada FAÇA**
          3. Suba um degrau da escada
        - **ENQUANTO a lâmpada não estiver livre do soquete FAÇA**
          6. Gire a lâmpada queimada no sentido anti-horário
        7. Posicione a lâmpada nova no soquete
        - **ENQUANTO a lâmpada não estiver firme no soquete FAÇA**
          6. Gire a lâmpada no sentido horário
        7. Desça da escada
        8. Guarde a escada

# Formas de representação de algoritmos

- **Descrição narrativa**
  - **Exercício 2:** elabore um algoritmo em um descrição narrativa para fritar um ovo



# Formas de representação de algoritmos

- **Descrição narrativa**

- **Exercício 2:** elabore um algoritmo em um descrição narrativa para fritar um ovo
  - Algoritmo
    1. Coloque um ovo na frigideira
    2. Espere o ovo ficar frito
    3. Remove o ovo da frigideira

# Formas de representação de algoritmos

- **Descrição narrativa**

- **Exercício 2:** elabore um algoritmo em um descrição narrativa para fritar um ovo
  - Algoritmo
    1. Coloque um ovo na frigideira
    2. Espere o ovo ficar frito
    3. Remove o ovo da frigideira

Como melhorar o algoritmo?

# Formas de representação de algoritmos

- **Descrição narrativa**

- **Exercício 2:** elabore um algoritmo em um descrição narrativa para fritar um ovo

- Algoritmo

1. Pegue a frigideira, espátula, ovo, óleo e sal
2. Coloque o óleo na frigideira
3. Acenda o fogo
4. Coloque a frigideira no fogo
5. Espere o óleo esquentar
6. Quebre o ovo
7. Despeje o ovo no óleo quente
8. Coloque o sal
9. Retire quando estiver pronto
10. Desligue o fogo



# Formas de representação de algoritmos

- **Descrição narrativa**
  - **Exercício 3:** elabore um algoritmo em um descrição narrativa que efetue a operação de envio de mensagem de texto via WhatsApp em um celular



# Formas de representação de algoritmos

- **Descrição narrativa**

- **Exercício 3:** elabore um algoritmo em um descrição narrativa que efetue a operação de envio de mensagem de texto via WhatsApp em um celular

- Algoritmo

1. Efetue desbloqueio do celular
2. Abra o aplicativo WhatsApp
3. Busque o contato a ser enviado a mensagem de texto
4. Selecione o contato
5. Selecione a barra para a escrita do texto
6. Digite o texto
7. Pressione o botão de envio
8. **SE o status da mensagem for no mínimo 1 traço cinza ENTÃO**
  9. Mensagem enviada com sucesso
9. **SENÃO**
  9. Mensagem não enviada

# Formas de representação de algoritmos

- **Fluxograma**

- Analisa o enunciado do problema e escreve utilizando símbolos gráficos para representar algoritmos
- Este tipo de representação existem símbolos padronizados para
  - Início
  - Fim
  - Fluxo de dados
  - Operação de entrada
  - Operação de saída
  - Operação de atribuição
  - Decisão

- **Vantagens**







- Maior clareza no fluxo de execução
- Linguagem visual
- Padrão mundial

- **Desvantagens**

- Requer conhecimento de convenções gráficas
- Mais trabalhoso em decorrência de seus desenhos
- Dificuldade para fazer correções
- Complica-se à medida que o algoritmo cresce

# Formas de representação de algoritmos

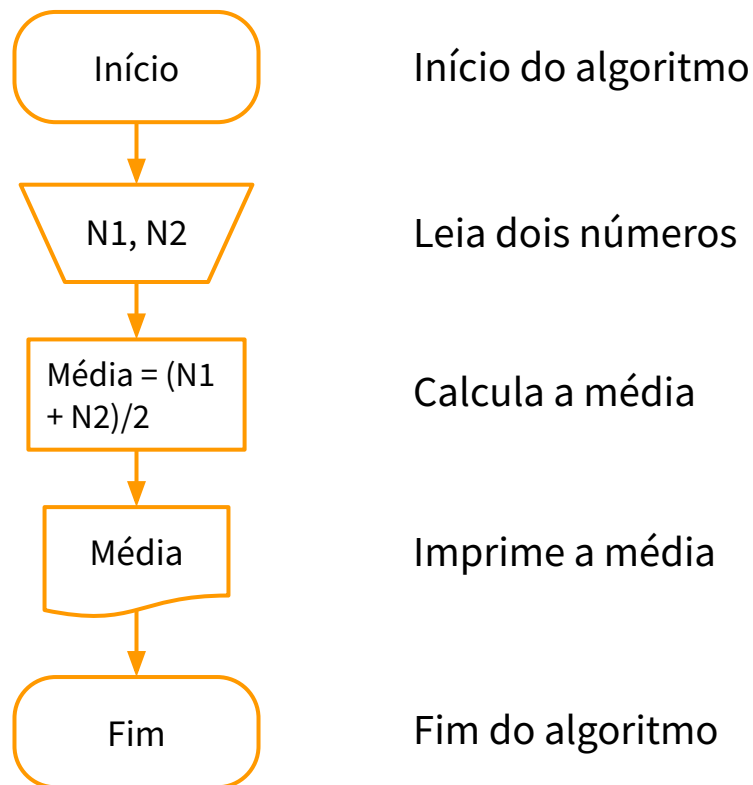
- **Fluxograma**
  - **Primitivas**

|  |                               |
|--|-------------------------------|
|    | Símbolo de início e fim       |
|    | Fluxo de dados                |
|    | Operação de entrada de dados  |
|   | Operação de saída de dados    |
|  | Operação de atribuição        |
|  | Operação de tomada de decisão |

# Formas de representação de algoritmos

- **Fluxograma**

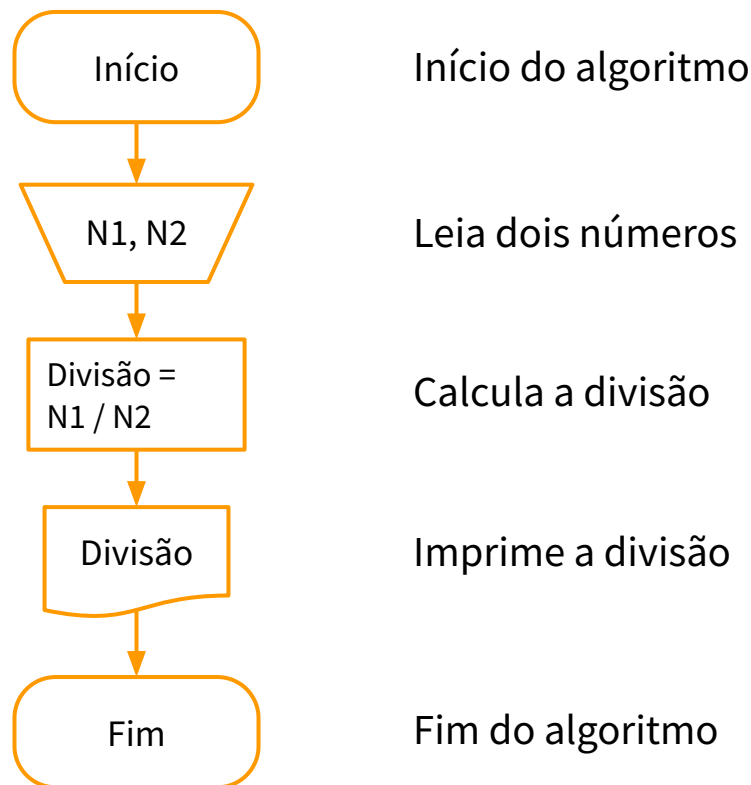
- **Exemplo 1:** calcule a média de dois números quaisquer



# Formas de representação de algoritmos

- **Fluxograma**

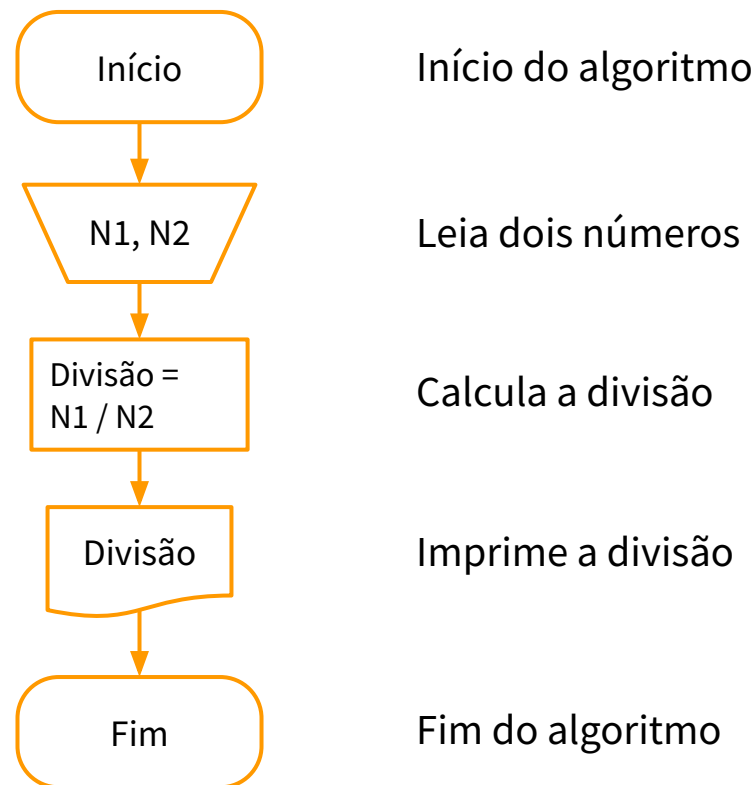
- **Exemplo 2:** calcule a divisão entre dois números



# Formas de representação de algoritmos

- **Fluxograma**

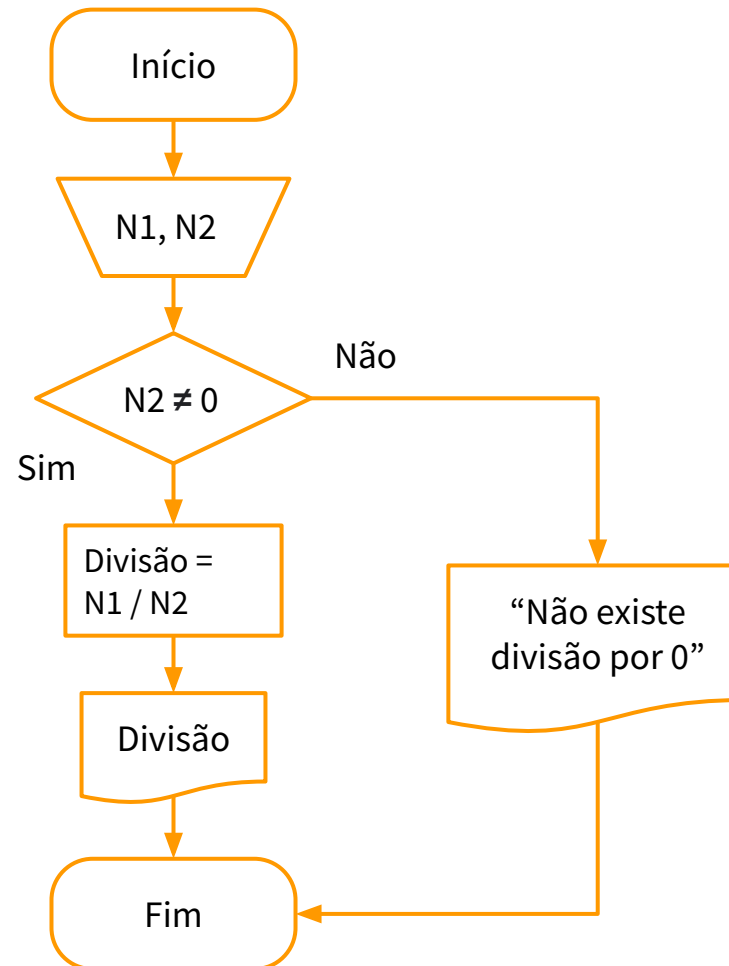
- **Exemplo 2:** calcule a divisão entre dois números



Vamos antecipar falhas:  
tratamento de erros

# Formas de representação de algoritmos

- Fluxograma
  - Exemplo 2



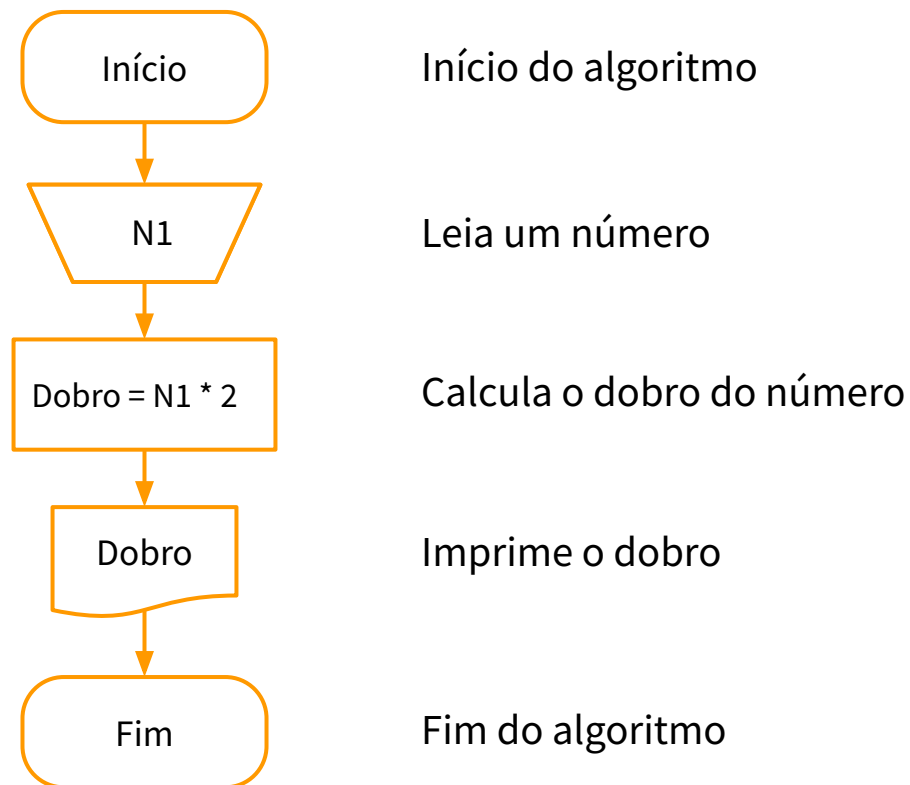


# Formas de representação de algoritmos

- **Fluxograma**
  - **Exemplo 3:** calcule o dobro de um número

# Formas de representação de algoritmos

- **Fluxograma**
  - **Exemplo 3:** calcule o dobro de um número



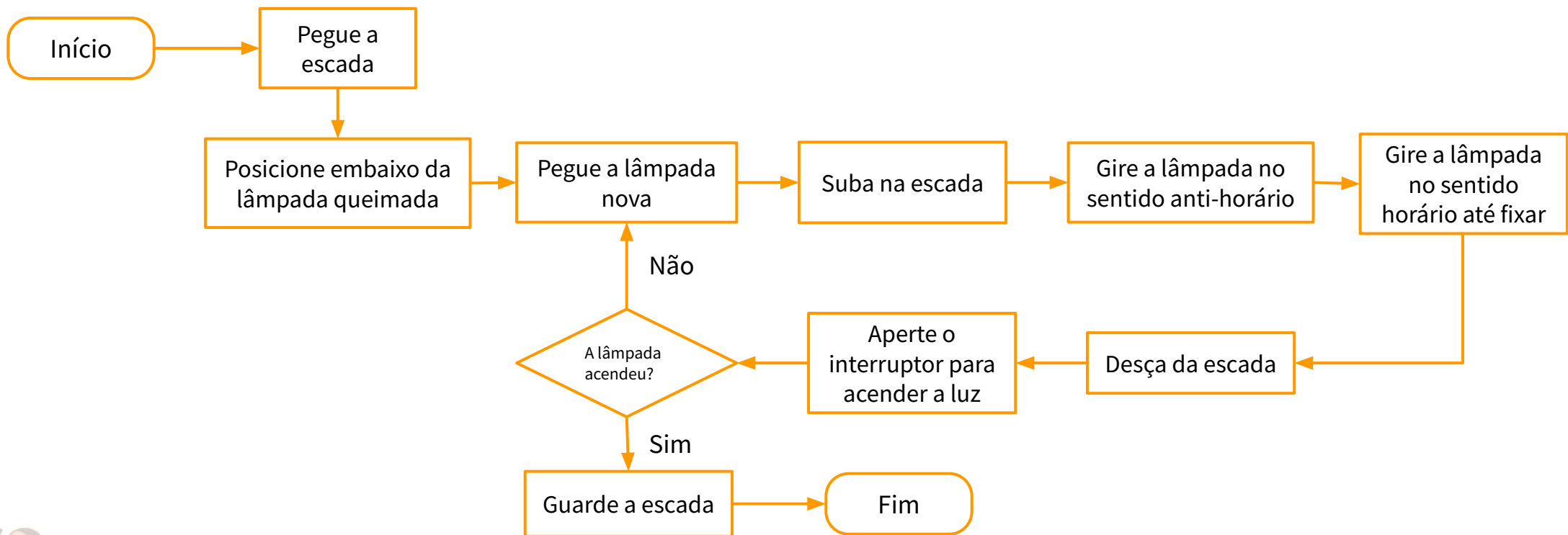
# Formas de representação de algoritmos

- **Fluxograma**
  - **Exemplo 4:** troque uma lâmpada

# Formas de representação de algoritmos

- **Fluxograma**

- **Exemplo 4:** troque uma lâmpada



# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- Vimos que a descrição narrativa pode ser interpretada de diversas maneiras, o que pode gerar ambiguidades. O fluxograma, por outro lado, tem maior precisão, mas não é muito descritivo, o que pode torná-lo insuficiente, além de se tornar mais complicado à medida que o algoritmo cresce
- O pseudocódigo é uma combinação das melhores características das duas formas de representação anteriores, em que as ideias podem ser expressas informalmente, mas obedecendo a regras de estrutura predefinidas para descrever um algoritmo
- Consiste na definição de uma pseudolinguagem de programação, cujo os comandos são em português, para representar algoritmos

# Formas de representação de algoritmos

- **Pseudocódigo ou portugal**

- Assemelhar-se bastante à forma em que os programas são escritos → bastante aceita
- Forma didática/pedagogia de introdução aos conceitos de linguagem de programação

# Formas de representação de algoritmos

- **Pseudocódigo ou portugal**

- **Vantagens**

- A principal é que mesmo sendo independente de qualquer linguagem de programação, sua estruturação facilita a transcrição do algoritmo criado para o código das linguagens de programação

- **Desvantagens**

- Exige o aprendizado das regras do pseudocódigo
    - Não padronização de sua estruturação. Isso significa que você encontrará o mesmo termo descrito de maneiras diferentes em diferentes literaturas

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- **Tipos de dados**

- Dados manipulados por um algoritmo podem possuir natureza distinta
      - Podem ser números, letras, frases, etc
    - Os tipos de dados surge para fazer a distinção entre dados de naturezas distintas
    - Por exemplo, o português estruturado (portugol) que utilizaremos para descrever nossos algoritmos possui o tipo de dado **inteiro**:
      - Consiste no conjunto de todos os números inteiros, denotado por **Z**, e todas as operações que podem ser aplicadas aos números inteiros (isto é, adição, subtração, multiplicação, divisão inteira e resto)



# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- **Tipos de dados**

- **Inteiro**

- Consiste dos números inteiros e das operações de adição, subtração, multiplicação, divisão inteira e resto
      - Os números inteiros podem ser negativos ou positivos, e ocupam 2 bytes
      - Por exemplo:
        - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 50, 100...
        - Números negativos são representados com o sinal “-” na frente do número, tal como -23

# Formas de representação de algoritmos

- **Pseudocódigo ou português**

- **Tipos de dados**

- **Real**

- Consiste dos números reais e das operações de adição, subtração, multiplicação, divisão
      - Os números reais são caracterizados por possuírem uma parte inteira e uma parte fracionária
      - Os números reais podem ser negativos ou positivos, e ocupam 4 bytes
      - Por exemplo
        - 3.141596
        - 1.78
        - -48.9

# Formas de representação de algoritmos

- **Pseudocódigo ou português**

- **Tipos de dados**

- **Caractere**

- Dados formados por um caractere, ou uma cadeia de caracteres
      - Inclui todas as letras, dígitos e caracteres de pontuação
      - Ocupa 1 byte de memória por caractere
      - Os elementos do conjunto de valores do tipo caractere devem ser escritos, nos algoritmos entre aspas duplas ou simples
        - “Luana Batista”
      - Há um elemento especial, “ ”, que é denominado de palavra vazia, pois não possui nenhum símbolo

# Formas de representação de algoritmos

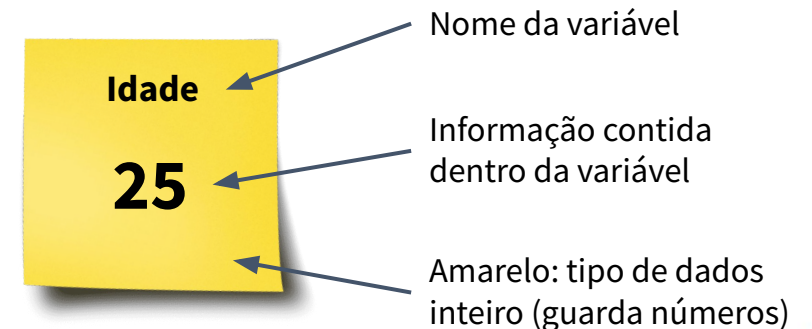
- **Pseudocódigo ou portugol**
  - **Tipos de dados**
    - **Lógico**
      - Também chamados de booleanos
      - Inclui apenas os valores lógicos falso e verdadeiro e as operações de negação (não), conjunção (e) e disjunção (ou)
      - Ocupam apenas 1 byte de memória

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- **Variável**

- Variáveis são criadas e podem ser imaginadas como uma “**caixa**” para armazenar valores de dados
    - Os dados são armazenados em locais específicos da memória, denominados **endereços de memória**
    - Cada variável representa uma posição de memória, e possui um **nome** e um **tipo de dado**
    - O conteúdo da variável pode **variar** ao longo do tempo, durante a execução de um programa



# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Componentes de uma variável**
    - **Nome**
      - Nome de uma variável deve ser único, isto é, identificar, de forma única, a variável no algoritmo
    - **Tipo de dado**
      - Define os valores que podem ser armazenados na variável
    - **Conteúdo**
      - Valor que ela armazena

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- **Variável**

- O ato de se criar uma variável é conhecido como declaração de variável
    - Variáveis são declaradas logo após **Var** e os tipos mais utilizados são inteiro, real, caractere e lógico

Em portugol, declaramos uma variável usando uma sentença da seguinte forma:

Var

**nome:** tipo

Por exemplo, a sentença:

Var

**salario:** real

Pode-se declarar mais de uma variável do mesmo tipo em uma mesma linha. Por exemplo,

Var

**salario, media:** real

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- **Identificadores**

- São nomes usados para designar os diferentes objetos existentes no programa, como por exemplo, o nome do programa, os tipos de dados e variáveis
    - É um nome único, não deve repetir no algoritmo
    - Regras básicas para formação
      - Caracteres permitidos: números, letras maiúsculas ou minúsculas e o underline “ \_ ”
      - O primeiro caractere deve ser sempre uma letra ou underline
      - Não são permitidos caracteres em branco e caracteres especiais (@, \$, +, -, %, !)
      - Não é possível utilizar palavras reservadas nos identificadores, ou seja, palavras que pertencem à linguagem de programação utilizada
      - Exemplos: a, A, Nota, nota, nota\_1, dia



# Formas de representação de algoritmos

- **Pseudocódigo ou português**
  - Operadores relacionais e lógicos

| Operador       | Símbolo | Exemplo           |
|----------------|---------|-------------------|
| Igual          | =       | $a = b$           |
| Diferente      | $\neq$  | $a \neq b$        |
| Maior          | >       | $a > b$           |
| Maior ou igual | $\geq$  | $a \geq b$        |
| Menor          | <       | $a < b$           |
| Menor ou igual | $\leq$  | $a \leq b$        |
| Conjunção      | e       | $a \text{ e } b$  |
| Disjunção      | ou      | $a \text{ ou } b$ |
| Negação        | não     | $\text{não } c$   |

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- **VisualG**

- O Visualg é um programa que permite criar, editar, interpretar e que também executa os algoritmos em português estruturado (portugol) como se fosse um “programa” normal de computador
    - Ideia: é um programa de livre uso e distribuição GRÁTIS, e DOMÍNIO PÚBLICO, usado para o ensino de lógica de programação em várias escolas e universidades no Brasil e no exterior

VisuALG



# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- Estrutura básica para algoritmos em pseudocódigo usando o VisualG

```
1 Algoritmo "semnome"  
2 // Disciplina: Introdução à Programação  
3 // Professora: Luana Batista da Cruz  
4  
5  
6 Var  
7 // Seção de Declarações das variáveis  
8  
9  
10 Inicio  
11 // Seção de Comandos, procedimento, funções, operadores, etc...  
12  
13  
14 Fimalgoritmo
```

**Início do código** "semnome" é o nome do algoritmo, não tem influência sobre o código

**Comentários** não são obrigatórios, mas ajudam no entendimento do código. No visualG usa-se // para iniciar um comentário

Nessa área as variáveis devem ser declaradas

Nessa área deve vir o código

**Fim do código**

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- **Sentença de atribuição**

- Tem a função de atribuir um determinado valor a uma variável declarada
    - Representada pelo símbolo <-
    - Ex:
      - `x <- 3`
      - `salario <- 1.212`
      - `bonificacao <- salario * 0.1`
      - `aula <- "Introdução à Programação"`
      - `teste <- FALSO`

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- **Sentença de atribuição**

- Tem a função de atribuir um determinado valor a uma variável declarada
    - Representada pelo símbolo <-
    - Ex:
      - `x <- 3`
      - `salario <- 1.212`
      - `bonificacao <- salario * 0.1`
      - `aula <- "Introdução à Programação"`
      - `teste <- FALSO`

Quais são os tipos de dados apropriado para cada uma das variáveis?

# Formas de representação de algoritmos

- Pseudocódigo ou portugol
  - Sentença de atribuição

```
1 Algoritmo "soma"  
2  
3 Var  
4   x, y, soma: inteiro  
5  
6 Inicio  
7   x <- 5  
8   y <- 7  
9   soma <- x + y  
10  
11 Fimalgoritmo
```

```
1 Algoritmo "simbolos"  
2  
3 Var  
4   simb1, simb2: caractere  
5  
6 Inicio  
7   simb1 <- "$"  
8   simb2 <- "@"  
9  
10 Fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Primitivas de entrada e saída**
    - **Entrada**
      - O comando de entrada é utilizado para receber dados informados pelo usuário
      - Representado pela palavra **leia**
      - Exemplo
        - leia(x) (os dados informados pelo usuário serão armazenados na variável x)

# Formas de representação de algoritmos

- Pseudocódigo ou português
  - Primitivas de entrada e saída
    - Entrada

```
1 Algoritmo "soma"  
2  
3 Var  
4   x, y, soma: inteiro  
5  
6 Inicio  
7   leia(x)  
8   leia(y)  
9   soma <- x + y  
10  
11 Fimalgoritmo
```



# Formas de representação de algoritmos

- **Pseudocódigo ou português**
  - **Primitivas de entrada e saída**
    - **Saída**
      - O comando de saída é utilizado para mostrar dados de uma variável ou conteúdo para o usuário, na tela do monitor, ou na impressora, entre outros
      - Representado pela palavra **escreva**
      - Exemplo
        - escreva(y) (mostra o valor armazenado na variável y)
        - escreva("O conteúdo de x é: ", x)

# Formas de representação de algoritmos

- Pseudocódigo ou portugol
  - Primitivas de entrada e saída
    - Saída

```
1 Algoritmo "soma"  
2  
3 Var  
4   x, y, soma: inteiro  
5  
6 Inicio  
7   escreva("Escreva o valor de x: ")  
8   leia(x)  
9   escreva("Escreva o valor de y: ")  
10  leia(y)  
11  soma <- x + y  
12  escreva("O resultado da soma é: ", soma)  
13  
14 Fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- **Pratique:** quais dos seguintes nomes são válidos como nomes de variáveis?
  - a) xyz\_2
  - b) \_
  - c) \_\_\_\_\_
  - d) x123
  - e) 123y
  - f) 1\_2
  - g) numero 1
  - h) fru?ta

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- **Pratique:** quais dos seguintes nomes são válidos como nomes de variáveis?

a) **xyz\_2**

b) **\_**

c) **\_\_\_\_\_**

d) **x123**

e) 123y

f) 1\_2

g) numero 1

h) fru?ta

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Pratique:** identifique o tipo de dados dos seguintes valores
    - a) “7 de setembro de 1822”
    - b) 1.3
    - c) falso
    - d) 31
    - e) “?”

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Pratique:** identifique o tipo de dados dos seguintes valores
    - a) “7 de setembro de 1822” - **caractere**
    - b) 1.3 - **real**
    - c) falso - **logico**
    - d) 31 - **inteiro**
    - e) “?” - **caractere**

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Pratique:** faça um algoritmo que receba seu nome e sobrenome e apresente na tela seu nome completo

# Formas de representação de algoritmos

- **Pseudocódigo ou português**

- **Pratique:** faça um algoritmo que receba seu nome e sobrenome e apresente na tela seu nome completo

```
1 Algoritmo "nome_sobrenome"
2
3 Var
4   nome: caractere
5   sobrenome: caractere
6
7 Inicio
8   escreva("Qual o seu nome? ")
9   leia(nome)
10  escreva("Qual o seu sobrenome? ")
11  leia(sobrenome)
12  escreva("Seu nome completo é: ")
13  escreva(nome, " ")
14  escreva(sobrenome)
15 Fimalgoritmo
```

```
1 Algoritmo "nome_sobrenome"
2
3 Var
4   nome: caractere
5   sobrenome: caractere
6
7 Inicio
8   escreva("Qual o seu nome? ")
9   leia(nome)
10  escreva("Qual o seu sobrenome? ")
11  leia(sobrenome)
12  escreva("Seu nome completo é: ")
13  escreva(nome, " ", sobrenome)
14 Fimalgoritmo
15
```



# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Exercício 1:** faça um programa que receba três notas, calcule e mostre a média aritmética entre elas

# Formas de representação de algoritmos

- **Pseudocódigo ou português**

- **Exercício 1:** faça um programa que receba três notas, calcule e mostre a média aritmética entre elas

```
1 Algoritmo "media_notas"
2
3 Var
4   n1, n2, n3, soma, media: real
5
6 Inicio
7   escreva("Digite as três notas: ")
8   leia(n1)
9   leia(n2)
10  leia(n3)
11  soma <- n1 + n2 + n3
12  media <- soma/3
13  escreva("A média das notas é: ", media)
14 Fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Exercício 2:** escreva um algoritmo que receba o salário de um funcionário, calcule e mostre o novo salário, sabendo-se que este sofreu um aumento de 25%

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- **Exercício 2:** escreva um algoritmo que receba o salário de um funcionário, calcule e mostre o novo salário, sabendo-se que este sofreu um aumento de 25%

```
1 Algoritmo "novo_salario"
2
3 Var
4     salario, novo_salario: real
5
6 Inicio
7     escreva("Digite o salário: ")
8     leia(salario)
9     novo_salario <- salario + (salario * 0.25)
10    escreva("Novo salário após aumento de 25%: ", novo_salario)
11 Fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Exercício 3:** escreva um algoritmo que receba três notas e seus respectivos pesos, calcule e mostre a média ponderada entre essas notas

# Formas de representação de algoritmos

- **Pseudocódigo ou português**

- **Exercício 3:** escreva um algoritmo que receba três notas e seus respectivos pesos, calcule e mostre a média ponderada entre essas notas

```
1 Algoritmo "media_ponderada"
2
3 Var
4   n1, n2, n3, p1, p2, p3, media_ponderada: real
5
6 Inicio
7   escreva("Digite a primeira nota: ")
8   leia(n1)
9   escreva("Digite o peso da primeira nota: ")
10  leia(p1)
11  escreva("Digite a segunda nota: ")
12  leia(n2)
13  escreva("Digite o peso da segunda nota: ")
14  leia(p2)
15  escreva("Digite a terceira nota: ")
16  leia(n3)
17  escreva("Digite o peso da terceira nota: ")
18  leia(p3)
19  media_ponderada <- (n1*p1 + n2*p2 + n3*p3) / (p1+p2+p3)
20  escreva("A média ponderada das notas é: ", media_ponderada)
21 Fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou português**

- **Sentença de condição**

- Altera o fluxo de execução de um programa baseado no valor (verdadeiro ou falso) de uma condição

```
se <condição> então  
    <executa as instruções caso a condição resulte em VERDADEIRO>  
senão  
    <executa as instruções caso a condição resulte em FALSO>  
fimse
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- **Sentença de condição**

- Altera o fluxo de execução de um programa baseado no valor (verdadeiro ou falso) de uma condição

```
se <condição> então
    <executa as instruções caso a condição resulte em VERDADEIRO>
senão
    se <condição> então
        <executa as instruções caso a condição resulte em VERDADEIRO>
    senão
        <executa as instruções caso a condição resulte em FALSO>
fimse
fimse
```



# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- **Sentença de condição**

- Altera o fluxo de execução de um programa baseado no valor (verdadeiro ou falso) de uma condição

```
1 Algoritmo "valor_negativo_positivo"
2
3 Var
4   n: real
5
6 Inicio
7   escreva("Digite um valor: ")
8   leia(n)
9   se (n < 0) então
10     escreva("Valor negativo!")
11   senão
12     escreva("Valor positivo!")
13   fimse
14 Fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou português**

- **Sentença de condição**

- Altera o fluxo de execução de um programa baseado no valor (verdadeiro ou falso) de uma condição

```
1 Algoritmo "divisao"
2
3 Var
4   n1, n2, resultado: real
5
6 Inicio
7   escreva("Digite o primeiro valor: ")
8   leia(n1)
9   escreva("Digite o segundo valor: ")
10  leia(n2)
11  se (n2 <> 0) então
12    resultado <- n1/n2
13    escreva("Resultado da divisão: ", resultado)
14  senão
15    escreva("Não é possível dividir por zero!")
16  fimse
17 Fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Sentença de condição**
    - **Exercício 1:** verifique se um número informado pelo usuário é par ou ímpar

# Formas de representação de algoritmos

- **Pseudocódigo ou português**
  - **Sentença de condição**
    - **Exercício 1:** verifique se um número informado pelo usuário é par ou ímpar

```
1 Algoritmo "impar_par"
2
3 Var
4   n, resultado: real
5
6 Inicio
7   escreva("Digite um número: ")
8   leia(n)
9   resultado <- n % 2
10  se (resultado = 0) então
11    escreva("É par!")
12  senão
13    escreva("É ímpar!")
14  fimse
15 Fimalgoritmo
16
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Sentença de condição**
    - **Exercício 2:** simule um caixa eletrônico para sacar dinheiro. O caixa eletrônico verifica se o valor que desejamos sacar é menor que o saldo disponível. Assuma que há R\$ 1000 de saldo disponível para o saque

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- **Sentença de condição**

- **Exercício 2:** simule um caixa eletrônico para sacar dinheiro. O caixa eletrônico verifica se o valor que desejamos sacar é menor que o saldo disponível. Assuma que há R\$ 1000 de saldo disponível para o saque

```
1 Algoritmo "sacar_valor"
2
3 Var
4     saldo_disponivel, valor_saque: real
5
6 Inicio
7     saldo_disponivel <- 1000
8     escreva("Informe o valor de saque: ")
9     leia(valor_saque)
10    se (valor_saque > saldo_disponivel) entao
11        escreval("O valor solicitado é maior que o valor disponível!")
12    senão
13        saldo_disponivel <- saldo_disponivel - valor_saque
14        escreval("Sacando R$ ", valor_saque, "!")
15    fimse
16    escreva("Saldo disponível: ", saldo_disponivel)
17 Fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Sentença de condição**
    - **Exercício 3:** dois números inteiros e encontrar o maior deles

# Formas de representação de algoritmos

- Pseudocódigo ou português
  - Sentença de condição
    - **Exercício 3:** dois números inteiros e encontrar o maior deles

```
1 Algoritmo "maior_numero"
2
3 Var
4   n1, n2: real
5
6 Inicio
7   escreva("Informe o primeiro número: ")
8   leia(n1)
9   escreva("Informe o segundo número: ")
10  leia(n2)
11  se (n1 > n2) então
12    escreva(n1, " é o maior!")
13  senão
14    se (n1 < n2) então
15      escreva(n2, " é o maior!")
16    senão
17      escreva("São iguais!")
18  fimse
19  fimse
20 Fimalgoritmo
```



# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Sentença de condição**
    - Operadores lógicos de disjunção, conjunção e negação
      - **Disjunção:** será verdadeira quando pelo menos uma das sentenças for verdadeira (ou)
      - **Conjunção:** será verdadeira somente quando todas as sentenças forem verdadeira (e)
      - **Negação:** terá valor falso quando a sentença for verdadeira e vice-versa (não)

# Formas de representação de algoritmos

- **Pseudocódigo ou portugal**

- **Sentença de condição**

- **Disjunção (ou)**

- Para entender esse operador vamos imaginar o seguinte exemplo: imagine que há gratuidade no transporte público para crianças com até 5 anos de idade e idosos a partir de 60 anos. Peça a idade do usuário e diga se ele tem direito ou não a gratuidade
      - Perceba que temos dois testes a serem feitos para descobrir quem tem gratuidade:
        - Idade menor que 6
        - Idade maior que 59

# Formas de representação de algoritmos

- Pseudocódigo ou portugol
  - Sentença de condição
    - **Disjunção (ou)**

```
1 Algoritmo "disjuncao"
2
3 Var
4   idade: inteiro
5
6 Inicio
7   escreva("Digite sua idade: ")
8   leia(idade)
9
10  se ((idade < 6) ou (idade > 59)) entao
11    escreva("Tem direito a gratuidade!")
12  senao
13    escreva("Não tem direito a gratuidade!")
14  fimse
15
16 Fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugal**

- **Sentença de condição**

- **Conjunção (e)**

- Para entender esse operador vamos imaginar o seguinte exemplo: imagine que há gratuidade no transporte público para crianças com até 5 anos de idade e idosos a partir de 60 anos. Peça a idade do usuário e diga se ele tem direito ou não a gratuidade
      - Perceba que nosso problema pode ser avaliado de outro ponto de vista, no caso descobrir quem não tem gratuidade:
        - Idade maior que 5
        - Idade menor que 60

# Formas de representação de algoritmos

- Pseudocódigo ou português
  - Sentença de condição
    - **Conjunção (e)**

```
1 Algoritmo "conjuncao"
2
3 Var
4   idade: inteiro
5
6 Inicio
7   escreva("Digite sua idade: ")
8   leia(idade)
9
10  se ((idade > 5) e (idade < 60)) entao
11    escreva("Não tem direito a gratuidade!")
12  senao
13    escreva("Tem direito a gratuidade!")
14  fimse
15
16 Fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugal**

- **Sentença de condição**

- **Negação (não)**

- Para entender esse operador vamos imaginar o seguinte exemplo: imagine que há gratuidade no transporte público para crianças com até 5 anos de idade e idosos a partir de 60 anos. Peça a idade do usuário e diga se ele tem direito ou não a gratuidade
      - Este operador é mais simples. Basicamente ele nega o resultado de algum teste, ou seja, se o resultado lógico de um teste for verdadeiro, ao negar teremos como resultado falso. De forma semelhante, se o resultado de um teste for falso, ao negar teremos como resultado verdadeiro

# Formas de representação de algoritmos

- Pseudocódigo ou portugol
  - Sentença de condição
    - **Negação (não)**

```
1 Algoritmo "negacao"
2
3 Var
4   idade: inteiro
5
6 Inicio
7   escreva("Digite sua idade: ")
8   leia(idade)
9
10  se (não((idade > 5) e (idade < 60))) entao
11    escreva("Tem direito a gratuidade!")
12  senao
13    escreva("Não tem direito a gratuidade!")
14  fimse
15
16 Fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Sentença de condição (operadores lógicos)**
    - **Exercício 1:** classifique os triângulos de acordo com o tamanho de seus lados (equiláteros, escalenos e isósceles)
      - Equilátero: todos os lados iguais
      - Escaleno: os três lados têm medidas diferentes
      - Isósceles: dois lados de mesma medida



# Formas de representação de algoritmos

- **Pseudocódigo ou português**
  - **Sentença de condição (operadores lógicos)**
    - **Exercício 1:** classifique os triângulos de acordo com o tamanho de seus lados (equiláteros, escalenos e isósceles)

```
1 Algoritmo "classificacao_triangulo"
2
3 Var
4   n1, n2, n3: real
5
6 Inicio
7   escreva("Digite o primeiro valor: ")
8   leia(n1)
9   escreva("Digite o segundo valor: ")
10  leia(n2)
11  escreva("Digite o terceiro valor: ")
12  leia(n3)
13
14  se ((n1 = n2) e (n2 = n3)) entao
15    escreval("Equilatero") // todos os lados iguais
16  senao
17    se ((n1 <> n2) e (n2 <> n3) e (n1 <> n3)) entao
18      escreval("Escalaeno") // os três lados com medidas diferentes
19    senao
20      escreval("Isóceles") // dois lados de mesma medida
21    fimse
22  fimse
23 Fimalgoritmo
```

# Formas de representação de algoritmos

- Qual é o mais legível?

```
1 Algoritmo "negacao"
2
3 Var
4   idade: inteiro
5
6 Inicio
7   escreva("Digite sua idade: ")
8   leia(idade)
9
10  se (não((idade > 5) e (idade < 60))) entao
11    escreva("Tem direito a gratuidade!")
12  senao
13    escreva("Não tem direito a gratuidade!")
14  fimse
15
16 Fimalgoritmo
```

```
1 Algoritmo "negacao"
2 Var
3   idade: inteiro
4 Inicio
5   escreva("Digite sua idade: ")
6   leia(idade)
7   se (não((idade > 5) e (idade < 60))) entao
8     escreva("Tem direito a gratuidade!")
9   senao
10    escreva("Não tem direito a gratuidade!")
11  fimse
12 Fimalgoritmo
13
14
15
```

# Formas de representação de algoritmos

- Qual é o mais legível?
  - **Indentação de código:** é empregada com o objetivo de ressaltar a estrutura do algoritmo (de forma hierárquica), aumentando assim a legibilidade, visualização e entendimento do código

```
1 Algoritmo "negacao"
2
3 Var
4   idade: inteiro
5
6 Inicio
7   escreva("Digite sua idade: ")
8   leia(idade)
9
10  se (não((idade > 5) e (idade < 60))) entao
11    escreva("Tem direito a gratuidade!")
12  senao
13    escreva("Não tem direito a gratuidade!")
14  fimse
15
16 Fimalgoritmo
```

```
1 Algoritmo "negacao"
2 Var
3   idade: inteiro
4 Inicio
5   escreva("Digite sua idade: ")
6   leia(idade)
7   se (não((idade > 5) e (idade < 60))) entao
8     escreva("Tem direito a gratuidade!")
9   senao
10    escreva("Não tem direito a gratuidade!")
11  fimse
12 Fimalgoritmo
13
14
15
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Sentença de repetição**
    - Três estruturas de repetição usuais nas linguagens de programação
    - O laço contado (repete uma sequência de comandos um determinado número de vezes)
    - E os laços condicionados (enquanto...faca e repita...ate)

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- **Sentença de repetição**

- Três estruturas de repetição usuais nas linguagens de programação
    - O laço contado (repete uma sequência de comandos um determinado número de vezes)

```
para <variável> de <valor-inicial> ate <valor-limite> [passo <incremento>] faça  
    <sequência-de-comandos>  
fimpara
```

# Formas de representação de algoritmos

- **Pseudocódigo ou português**
  - **Sentença de repetição**
    - Para.. até.. faça

```
1 Algoritmo "num_1_10"  
2  
3 Var  
4   j: inteiro  
5  
6 Inicio  
7   para j de 1 ate 10 faça  
8     escreva (j)  
9   fimpara  
10  
11 Fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Sentença de repetição**
    - Para.. até.. faça

```
1 Algoritmo "num_10_1"  
2  
3 Var  
4   j: inteiro  
5  
6 Inicio  
7   para j de 10 ate 1 faça  
8     escreva (j)  
9   fimpara  
10  
11 Fimalgoritmo
```

E se eu inverter?

# Formas de representação de algoritmos

- **Pseudocódigo ou português**
  - **Sentença de repetição**
    - Para.. até.. faça

```
1 Algoritmo "num_10_1"  
2  
3 Var  
4   j: inteiro  
5  
6 Inicio  
7   para j de 10 ate 1 faça  
8     escreva (j)  
9   fimpara  
10  
11 Fimalgoritmo
```

E se eu inverter?  
**Não funciona!**



# Formas de representação de algoritmos

- **Pseudocódigo ou português**
  - **Sentença de repetição**
    - Para.. até.. faça

```
1 Algoritmo "num_10_1"  
2  
3 Var  
4   j: inteiro  
5  
6 Inicio  
7   para j de 10 ate 1 passo -1 faça  
8     escreva (j)  
9   fimpara  
10  
11 Fimalgoritmo
```

**Solução:** passo -1

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Sentença de repetição (Para.. até.. faça)**
    - **Exemplo 1:** leia um valor para uma variável N de 1 a 10 e calcule a tabuada de N

# Formas de representação de algoritmos

- **Pseudocódigo ou português**
  - **Sentença de repetição (Para.. até.. faça)**
    - **Exemplo 1:** leia um valor para uma variável N de 1 a 10 e calcule a tabuada de N

```
1 Algoritmo "tabuada"  
2  
3 Var  
4   n, j: inteiro  
5 Inicio  
6   escreval("Informe o valor de N: ")  
7   leia(n)  
8   escreval("A tabula de ",n, " é: ")  
9   para j de 0 ate 10 faça  
10      escreval(j, " x ", n, ": ", j*n)  
11   fimpara  
12 Fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Sentença de repetição (Para.. até.. faça)**
    - **Exemplo 2:** escreva os números ímpares entre 100 e 200

# Formas de representação de algoritmos

- **Pseudocódigo ou português**
  - **Sentença de repetição (Para.. até.. faça)**
    - **Exemplo 2:** escreva os números ímpares entre 100 e 200

```
1 algoritmo "Impares"
2
3 var
4   i: inteiro
5
6 inicio
7   para i de 100 ate 200 faça
8     se (i mod 2 <> 0) então
9       escreval(i)
10    fimse
11  fimpara
12 fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Sentença de repetição (Para.. até.. faça)**
    - **Exemplo 3:** calcule o fatorial de um número. Ex:  $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

# Formas de representação de algoritmos

- Pseudocódigo ou portugol
  - **Sentença de repetição (Para.. até.. faça)**
    - **Exemplo 3:** calcule o fatorial de um número. Ex:  $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

```
1 algoritmo "Fatorial"
2
3 var
4   n, i: inteiro
5   fatorial: inteiro
6
7 inicio
8   escreva("Digite um número para calcular o fatorial: ")
9   leia(n)
10  fatorial <- n
11  para i de n-1 ate 1 passo -1 faça
12    fatorial <- fatorial * i
13  fimpara
14  escreva("O fatorial de ", n, " é : ", fatorial)
15 fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou português**

- **Sentença de repetição**

- O laço condicionados (repete uma sequência de comandos enquanto uma determinada condição (especificada através de uma expressão lógica) for satisfeita)

```
enquanto <expressão-lógica> faça  
    <sequência-de-comandos>  
fimenquanto
```

```
repita  
    <sequência-de-comandos>  
ate <expressão-lógica>
```



# Formas de representação de algoritmos

- **Pseudocódigo ou português**
  - **Sentença de repetição**
    - Enquanto.. faça

```
1 Algoritmo "num_1_10"  
2  
3 Var  
4   j: inteiro  
5  
6 Inicio  
7   j <- 1  
8   enquanto j <= 10 faça  
9     escreva (j)  
10    j <- j + 1  
11  fimenquanto  
12  
13 Fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Sentença de repetição (Enquanto.. faça)**
    - **Exemplo 1:** leia uma quantidade não determinada de números positivos. Por fim, retorne a média dos valores pares e ímpares. O programa encerrará ao digitar -1

- Exemplo 1

```
1 algoritmo "Media_par_impar"
2
3 var
4     acumula_par, acumula_impar: real
5     contador_par, contador_impar, n: inteiro
6
7 inicio
8     acumula_par <- 0
9     acumula_impar <- 0
10    contador_par <- 0
11    contador_impar <- 0
12    escreva("Digite um valor: ")
13    leia(n)
14    enquanto (n <> -1) faça
15        se (n mod 2 = 0) então
16            acumula_par <- acumula_par + n
17            contador_par <- contador_par + 1
18        senão
19            acumula_impar <- acumula_impar + n
20            contador_impar <- contador_impar + 1
21        fimse
22        escreva("Digite um valor: ")
23        leia(n)
24    fimenquanto
25
26    se (contador_par > 0) então
27        escreval("Média dos par: ", acumula_par/contador_par)
28    fimse
29    se (contador_impar > 0) então
30        escreval("Média dos ímpares: ", acumula_impar/contador_impar)
31    fimse
32 finalgoritmo
33
```

# Formas de representação de algoritmos

- **Pseudocódigo ou português**
  - **Sentença de repetição (Enquanto.. faça)**
    - **Exemplo 2:** escreva um programa que retorne a sequência de Fibonacci até o limite informado
    - Na matemática, a sequência de Fibonacci é uma sequência de números inteiros, começando normalmente por 0 e 1, na qual cada termo subsequente corresponde à soma dos dois anteriores
      - Ex: Se a entrada for igual a 7. A sequência de Fibonacci será: 0 1 1 2 3 5 8

# Formas de representação de algoritmos

- Pseudocódigo ou portugol
  - Sentença de repetição (Enquanto.. faça)
    - Exemplo 2

```
1 Algoritmo "Fibonacci"
2
3 Var
4   n, aux, anterior, atual, contador: inteiro
5
6 Inicio
7   escreva("Digite o valor máximo para fibonacci: ")
8   leia(n)
9   anterior <- -1
10  atual <- 1
11  contador <- 0
12  enquanto (contador < n) faça
13    aux <- atual
14    atual <- atual + anterior
15    anterior <- aux
16    contador <- contador + 1
17    escreva(atual)
18  fimenquanto
19
20 Fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou português**
  - **Sentença de repetição**
    - Repita.. até

```
1 Algoritmo "num_1_10"  
2  
3 Var  
4   j: inteiro  
5  
6 Inicio  
7   j <- 1  
8   repita  
9     escreva (j)  
10    j <- j + 1  
11    ate j > 10  
12  
13 Fimalgoritmo
```

# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**
  - **Sentença de repetição (Repita.. até)**
    - **Exemplo 1:** leia a quantidade de alunos de uma turma. Posteriormente, para cada aluno calcule e mostre a média das três notas informadas. Por fim, mostre a média geral das notas dos alunos

# Formas de representação de algoritmos

- Pseudocódigo ou português
  - Sentença de repetição (Repita.. até)
    - Exemplo 1

```
1 Algoritmo "media_geral"
2
3 Var
4   n1, n2, n3, media, soma_media: real
5   n_alunos, contador: inteiro
6
7 Inicio
8   contador <- 0
9   escreva("Digite a quantidade de alunos: ")
10  leia(n_alunos)
11
12  repita
13    escreva("Digite a primeira nota: ")
14    leia(n1)
15    escreva("Digite a segunda nota: ")
16    leia(n2)
17    escreva("Digite a terceira nota: ")
18    leia(n3)
19    media <- (n1 + n2 + n3) / 3
20    escreval("Média: ", media)
21    soma_media <- soma_media + media
22    contador <- contador + 1
23  ate(contador = n_alunos)
24  escreva("A média geral das notas dos alunos é: ", soma_media/n_alunos)
25 Fimalgoritmo
```



# Formas de representação de algoritmos

- **Pseudocódigo ou portugol**

- **Sentença de repetição (Repita.. até)**

- **Exemplo 2:** a sequência de Fettuccine será obtida da seguinte forma: irá solicitar que o usuário digite os 2 primeiros termos e que também informe até qual termo deseja conhecer da sequência (deixar claro ao usuário que ele no mínimo deverá querer ver 3 termos)
    - Para obter a sequência de Fettuccine, os 2 primeiros termos são informados pelo usuário e a partir daí segue a seguinte regra: se o próximo termo for de posição ímpar, ele será obtido pela soma dos 2 anteriores; se o próximo termo for de posição par, ele será obtido pela subtração do primeiro anterior pelo segundo anterior
      - Ex: o usuário digitou os números 7 e 10, e informou que quer conhecer 6 termos. O resultado da sequência de Fettuccine é: 7, 10, 17, 7, 24, 17

- Exemplo 2

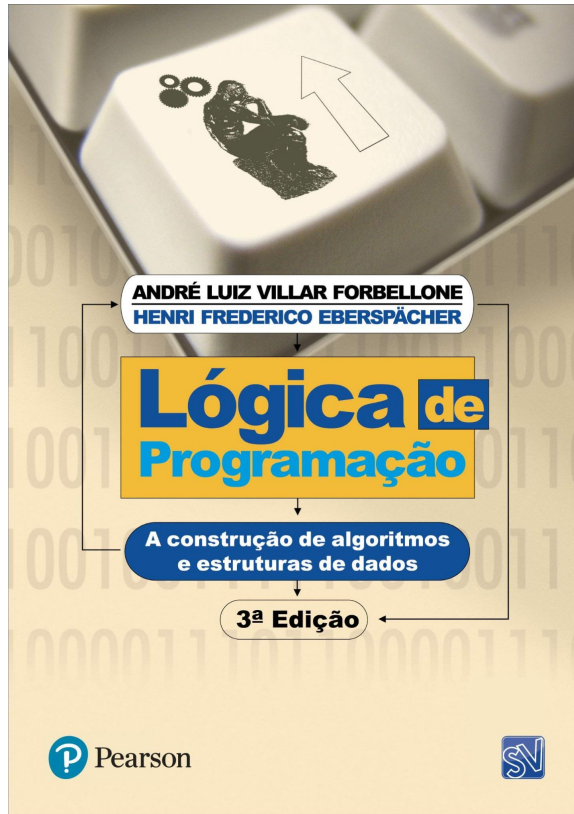
```
1 algoritmo "Fetuccine"
2
3 var
4     num1, num2, valor_termo, numero_termos, i: inteiro
5
6 inicio
7     i <- 3
8
9     escreval("Digite o primeiro termo: ")
10    leia(num1)
11    escreval("Digite o segundo termo: ")
12    leia(num2)
13
14    repita
15        escreval("Digite a quantidade de termos: ")
16        leia(numero_termos)
17    ate (numero_termos >= 3)
18
19    escreva(num1:3)
20    escreva(num2:3)
21
22    repita
23        se i mod 2 = 0 entao
24            valor_termo <- num2 - num1
25        senao
26            valor_termo <- num2 + num1
27        fimse
28
29        escreva(valor_termo:3)
30
31        num1 <- num2
32        num2 <- valor_termo
33        i <- i + 1
34
35    ate ( i > numero_termos )
36 fimalgoritmo
```

# Resumindo..

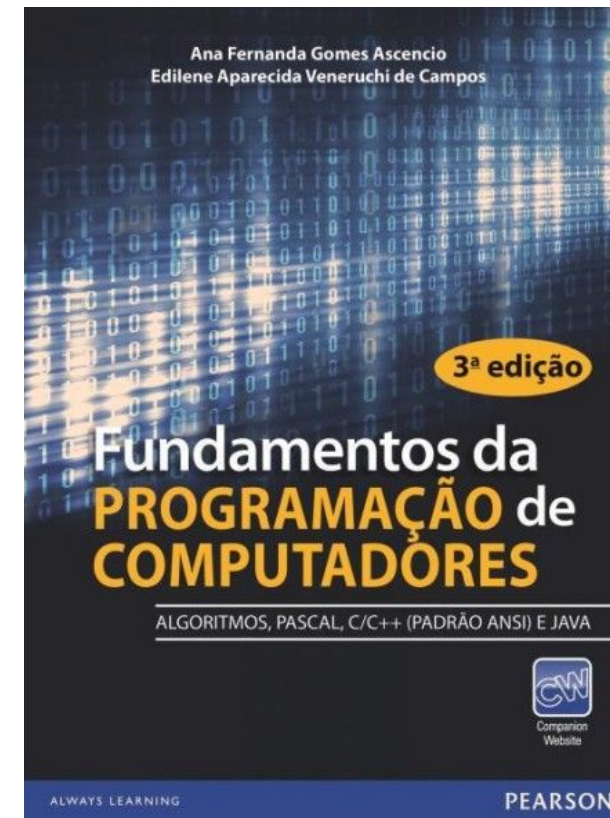
- Conceitos preliminares da computação
- Objetivo básico e finalidade da computação
- Formas de representação de algoritmos
  - Descrição narrativa
  - Fluxograma
  - Linguagem algorítmica, pseudocódigo ou português estruturado



# Referências

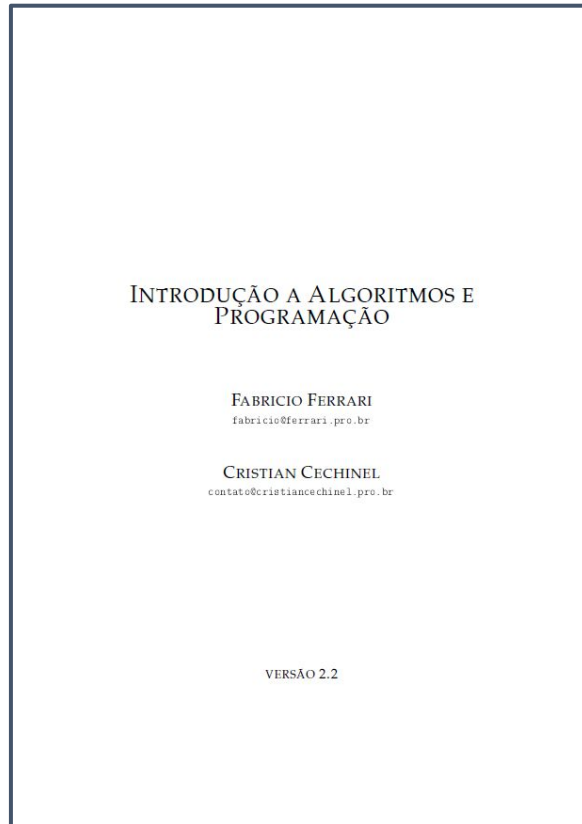


FORBELLONE, A.L.V., **Lógica de Programação. A Construção de Algoritmos e Estrutura de Dados**, Pearson, São Paulo, 2005.

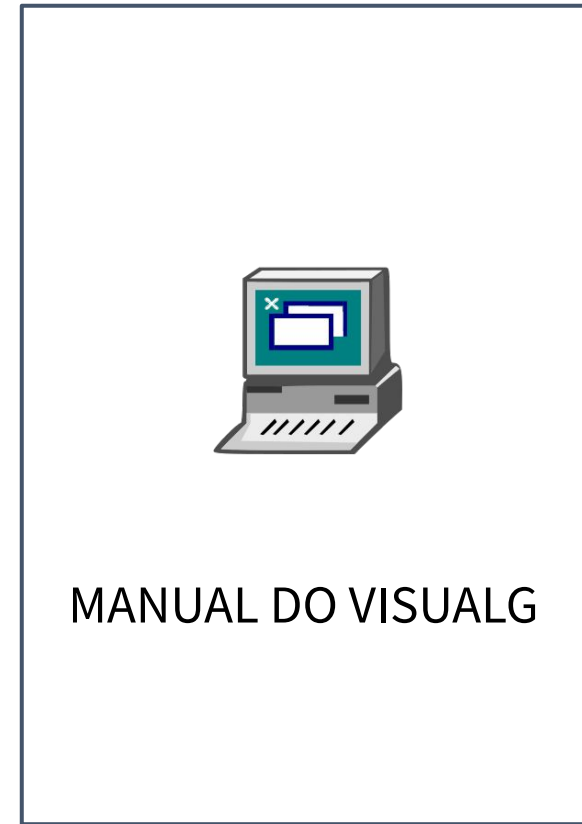


ASCENCIO, Ana F Gomes; CAMPOS, Edilene A. V. de. **Fundamentos de programação de computadores: algoritmos, Pascal e C/C++**. São Paulo: Prentice Hall, 2002

# Referências



FERRARI, Fabricio; CECHINEL, Cristian.  
**Introdução a Algoritmos e Programação.**  
Versão 2.2.



Manual do VisualG 3.0