

<http://rogerioaraujo.wordpress.com>

**Série Desenvolvimento de Sistemas**

# **Lógica de Programação e Estruturas de Dados**

**Rogério Araújo**

**Série Desenvolvimento de Sistemas**

# **Lógica de Programação e Estruturas de Dados**

**Rogério Araújo**

## O Autor

---

*O homem é aquilo que ele próprio faz.*

*André Malraux*

Nasci em Brasília/DF, porém, ainda pequeno, meus pais se mudaram para Teresina/PI, onde cresci com muito sol na moleira. Passei no concurso do Ministério Público da União (MPU) e fui, em janeiro de 2005, lotado inicialmente na Procuradoria da República do Estado Pará em Belém/PA, terra essa que quando não chove todo dia, chove o dia todo. Essa cidade me acolheu muito bem e fiz inúmeros amigos para levar para a vida toda e, além disso, ganhei várias mães por tabela. Atualmente, estou lotado na Procuradoria da República Federal (PGR), em Brasília/DF. Resido hoje nessa capital e conquistei novos grandes amigos tanto conhecidos no novo local de trabalho quanto nessa vida de concurseiro.



Sou especialista em Governança em TI pela Unieuro ([www.unieuro.edu.br](http://www.unieuro.edu.br)) e especialista em Desenvolvimento de Sistemas Baseados em Software Livre pela UNAMA (Universidade da Amazônia) ([www.unama.br](http://www.unama.br)), graduado no curso de Bacharelado em Ciência da Computação pela UESPI (Universidade Estadual do Piauí) ([www.uespi.br](http://www.uespi.br)) e conclui o Curso Técnico de Processamento de Dados pela ETEPI (Escola Técnica Estadual do Piauí). No ramo de certificações, possuo a COBIT 4.1 Foundation Certified e a SCJA (Sun Certified Associate for J2SE).

Mantenho o blog <http://rogerioaraujo.wordpress.com>. Escrevo posts sobre dicas e assuntos para os concursos de TI e sou autor de artigos no site do professor Walter Cunha ([www.waltercunha.com](http://www.waltercunha.com)).

Abraços e vamos nessa!

# Sumário

---

*O único lugar onde o sucesso vem antes do trabalho é no dicionário.*

*Albert Einstein*

Prefácio .....	4
----------------	---

## PARTE I – LÓGICA DE PROGRAMAÇÃO

Capítulo 1 Introdução .....	7
1.1 Para começar .....	7
1.1.1 Algoritmos, resolução de problemas e lógica .....	8
1.2 Algoritmos .....	10
1.3 Características de um algoritmo .....	11
1.4 Fases de um algoritmo .....	12
1.5 Método para construção de um algoritmo .....	12
1.6 Regras básicas para construção de um algoritmo .....	13
1.7 Estruturas que formam um algoritmo .....	13
1.8 Representações de um algoritmo .....	15
1.9 Forma geral de um algoritmo (pseudocódigo) .....	16
1.10 Itens importantes para um algoritmo (pseudocódigo) .....	17
1.10.1 Atribuições importantes .....	17
1.11 Variáveis e constantes .....	18
1.11.1 Analogia de variáveis e constantes com garagens de carros .....	18
1.11.2 Declaração de variáveis e constantes .....	19
1.11.3 Regras para nomes de variáveis e constantes .....	19
1.12 Algoritmo 1 – Encontrar o maior número de dois inteiros .....	20
1.13 Entrada e saída de dados .....	23
1.13.1 Entrada .....	23
1.13.2 Algoritmo 2 – Ler um número inteiro .....	23
1.13.3 Saída .....	24
1.13.4 Algoritmo 3 – Ler um número inteiro (modificado) .....	24
1.14 Teste de mesa ou teste chinês .....	25
1.14.1 Algoritmo 4 – Calcular salário a receber .....	25
1.15 Questões do capítulo .....	27
1.16 Respostas das questões do capítulo .....	29
1.17 Questões de concursos.....	36
1.18 Gabarito comentado das questões de concursos .....	39

Capítulo 2 Tipos de dados .....  
    2.1 Simples .....  
    2.2 Compostos .....  
Capítulo 3 O que é mais necessário saber .....  
    3.1 Avaliação de expressões .....  
    3.2 Operadores e expressões .....  
    3.3 Estruturas de controle de fluxo.....  
        3.3.1 Seleção .....  
        3.3.2 Repetição .....  
    3.4 Módulos .....  
    3.5 Recursividade .....  
Capítulo 5 Algoritmos .....  
    5.1 Busca .....  
    5.2 Pesquisa .....  
    5.3 Ordenação .....  
Capítulo 6 Referências .....

**PARTE II – ESTRUTURAS DE DADOS**

Capítulo 1 Pilhas .....  
Capítulo 2 Filas .....  
Capítulo 3 Listas lineares .....  
Capítulo 4 Árvores .....  
Capítulo 5 Grafos .....  
Capítulo 6 Referências .....

## Prefácio

---

*A melhor maneira de melhorar o padrão de vida está em melhorar o padrão de pensamento.*

*U. S. Andersen*

### Objetivo

Este material visa ser mais um item auxiliador na luta de vocês por uma vaga no concurso do seu sonho. Vou me esforçar para isso! 😊

Quero trazer uma linguagem informal e bem humorada para que os assuntos abordados sejam mais bem assimilados e que você não possa se cansar em lê-lo e sim ter o prazer em lê-lo.

O foco deste material é tentar ensinar como responder questões que abordem Lógica de Programação e Estruturas de Dados. **Não** vamos aprender a desenvolver algoritmos. Entretanto, teremos vários exemplos para nos familiarizarmos com algoritmos e estruturas de dados.

### Público-alvo

Este material é endereçado a todos àqueles que precisam se preparar para concursos que exijam conhecimentos em Lógica de Programação e Estruturas de Dados.

### Organização geral

Dividi o material em duas partes:

- Parte I: Lógica de Programação; e
- Parte II: Estruturas de Dados.

Utilizei editais de algumas bancas para trazer os itens de assuntos mencionados acima. Os itens serão organizados em capítulos dentro de sua respectiva parte.

Ao final de cada capítulo, teremos questões de minha autoria para revisão do que acabamos de ver e questões de concursos para sabermos como as bancas estão cobrando e como podemos resolvê-las.

### Informações de apoio

Para nos ajudar no entendimento das explicações, trago quadros explicativos dos tipos:

- **Quadros de notas importantes** (quadros de cor verde): para notas importantes que precisam de um destaque melhor;
- **Quadros de exemplos** (quadros de cor azul): para exemplos que enriquecerão o entendimento dos assuntos; e
- **Quadros de observações** (quadros de cor amarela): para observações gerais para explicações diversas sobre o curso.

Exemplos dos quadros de notas importantes:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque consequat placerat nibh, eu pulvinar lacus lacinia ut. Donec ultricies eleifend luctus. Mauris luctus bibendum sapien.

Exemplos dos quadros de exemplos:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque consequat placerat nibh, eu pulvinar lacus lacinia ut. Donec ultricies eleifend luctus. Mauris luctus bibendum sapien.

Exemplos dos quadros de observações:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque consequat placerat nibh, eu pulvinar lacus lacinia ut. Donec ultricies eleifend luctus. Mauris luctus bibendum sapien.

## Conclusão

Quero esclarecer que este material **não** é concorrente de outros que conhecemos na comunidade concurseira de TI. Ele pode e deve ser visto com um recurso disponível para ser usado em conjunto com os outros de mesma natureza.

**Não substitua os livros conceituados por qualquer material como este no mercado**, em hipótese alguma, ou seja, se for para decidir em comprar livros ou guias como este, compre livros! Oriente a vocês que vejam as referências aqui citadas para possa adquirir livros e acessar os sites.

Galera, espero que possa mais uma vez ajudá-los como venho tentando fazer com auxílio de meu blog. Minha maior recompensa é ser uma opção de qualidade para nossos estudos do dia a dia.

Por favor, aproveite este material e consiga seus objetivos! 😊

Simbora!!!

# PARTE I

## LÓGICA DE PROGRAMAÇÃO



# Introdução

*A persistência é o menor caminho do êxito.*

*Charles Chaplin*

## 1.1 Para começar

No nosso dia a dia, nos deparamos com várias situações onde precisamos chegar a um objetivo específico aplicando um conjunto finito de passos. Por exemplo, para trocar uma lâmpada queimada, precisamos comprar uma lâmpada nova, desligar a chave geral (quem vai querer pegar um choque?), pegar uma escada, tirar a queimada, colocar a nova, ligar a chave geral e ligar o interruptor da lâmpada. Vejam que o objetivo era trocar uma lâmpada e para isso traçamos passos para chegar ao resultado final. Bom, temos vários outros exemplos em nossas vidas, como chegar ao nosso trabalho pegando um melhor caminho, fazer um bolo, fazer uma faxina, organizar os livros, etc. Resumindo, o homem é um solucionador de “pepinos” (figura 1.1).

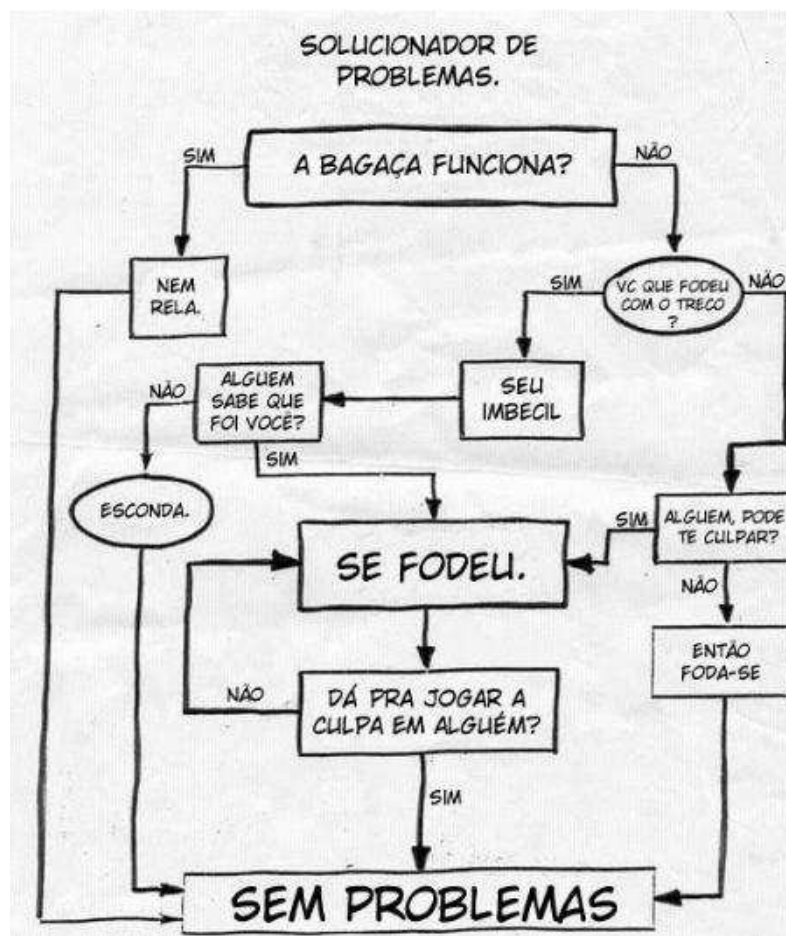


Figura 1.1: Solucionando problemas.

### 1.1.1 Algoritmos, resolução de problemas e lógica

Essa capacidade de resolução de problemas é também transportada para a construção de programas de computador (nota 1.1).

A **construção de programas de computador** exige, entre outras habilidades, a **capacidade de resolver problemas** através da **identificação de um conjunto ordenado e finito de etapas e/ou instruções** que levam a sua resolução.

Nota 1.1: Construção de programas de computador e resolução de problemas.

Para construirmos nossos programas de computador, precisamos entender a importância do conceito de **algoritmos**. Quero frisar que quando falamos de capacidade de resolução de problemas, estamos falando de algoritmos.

Quando falamos de **capacidade de resolução de problemas**, estamos falando de **algoritmos**.

Nota 1.2: Algoritmos e resolução de problemas.

O conceito de algoritmo não é de exclusividade da área da computação e, mais ainda, um algoritmo também não representa, necessariamente, um programa de computador e sim os passos necessários para realizar uma tarefa ou solucionar um problema, seja de que área for. Como citado bem no início, nos deparamos com vários problemas que exigem soluções e podemos usar algoritmos para resolvê-los.

O **conceito de algoritmo não é de exclusividade** da área da computação. Um algoritmo também **não representa**, necessariamente, um **programa de computador** e sim os passos necessários para realizar uma tarefa ou solucionar um problema, seja de que área for.

Nota 1.3: Algoritmos, área da computação e programas de computador.

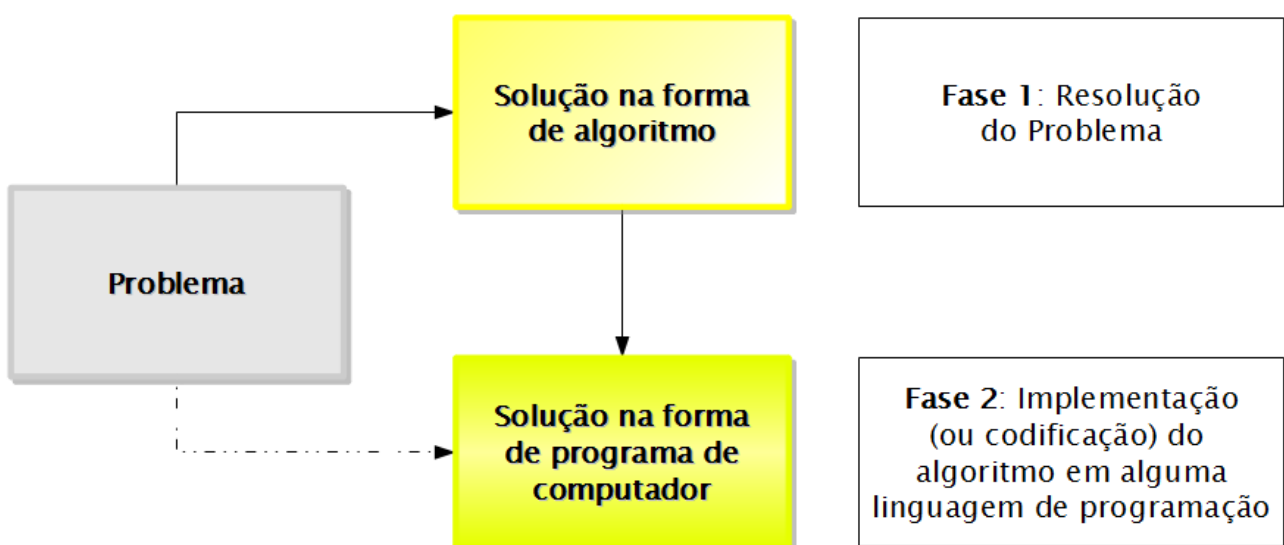


Figura 1.2: Problema, algoritmo e programa de computador.

Quando temos um problema para ser traduzido em programa de computador, um passo importante é resolver o problema na forma de algoritmo. A partir disso, já podemos implementar o algoritmo em alguma linguagem de programação (figura 1.2).

Bom, falamos um pouco de algoritmo e a capacidade de resolução de problemas, mas como encaixamos o conceito de lógica junto aos itens citados? O conceito de lógica nos dará justamente o poder de raciocínio de como resolver uma tarefa ou problema. Na nota 1.4, temos alguns conceitos sobre lógica e lógica específica de programação.

**Lógica** pode ser definida como sendo o estudo das leis do raciocínio e do modo de aplicá-las corretamente na demonstração da verdade (VENANCIO, 1997).

A utilização da **lógica** na vida do indivíduo é constante, visto que é ela quem possibilita a ordenação do pensamento humano (FORBELLONE, 1993).

A **lógica de programação** consiste em aprender a pensar na mesma sequência de execução dos programas de computador (ESMIN, 2000).

Nota 1.4: Conceitos de lógica e lógica de programação.

Para finalizarmos, uma revisãozinha dos conceitos vistos acima (figuras 1.3 e 1.4):

- Quando falamos de capacidade de resolução de problemas, estamos falando de algoritmos;
- A lógica nos dará o poder de raciocínio de como resolver uma tarefa ou problema;
- No caso de construção de um programa de computador, a lógica de programação consiste em aprender a pensar na mesma sequência de execução desse tipo de programa; e
- Um algoritmo não representa, necessariamente, um programa de computador, e sim os passos necessários para realizar uma tarefa ou solucionar um problema.

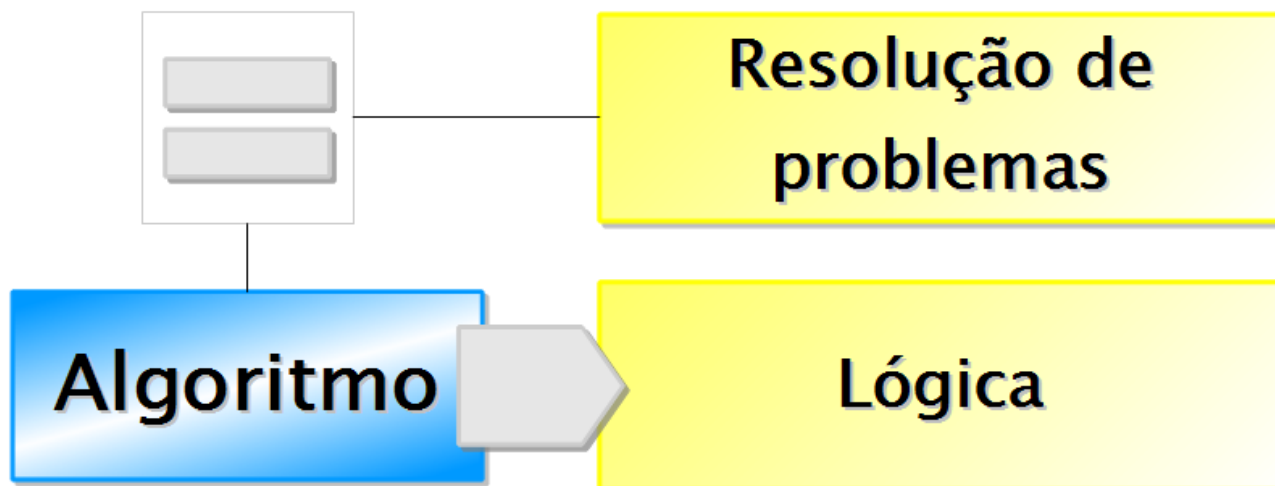


Figura 1.3: Algoritmo, resolução de problemas e lógica.

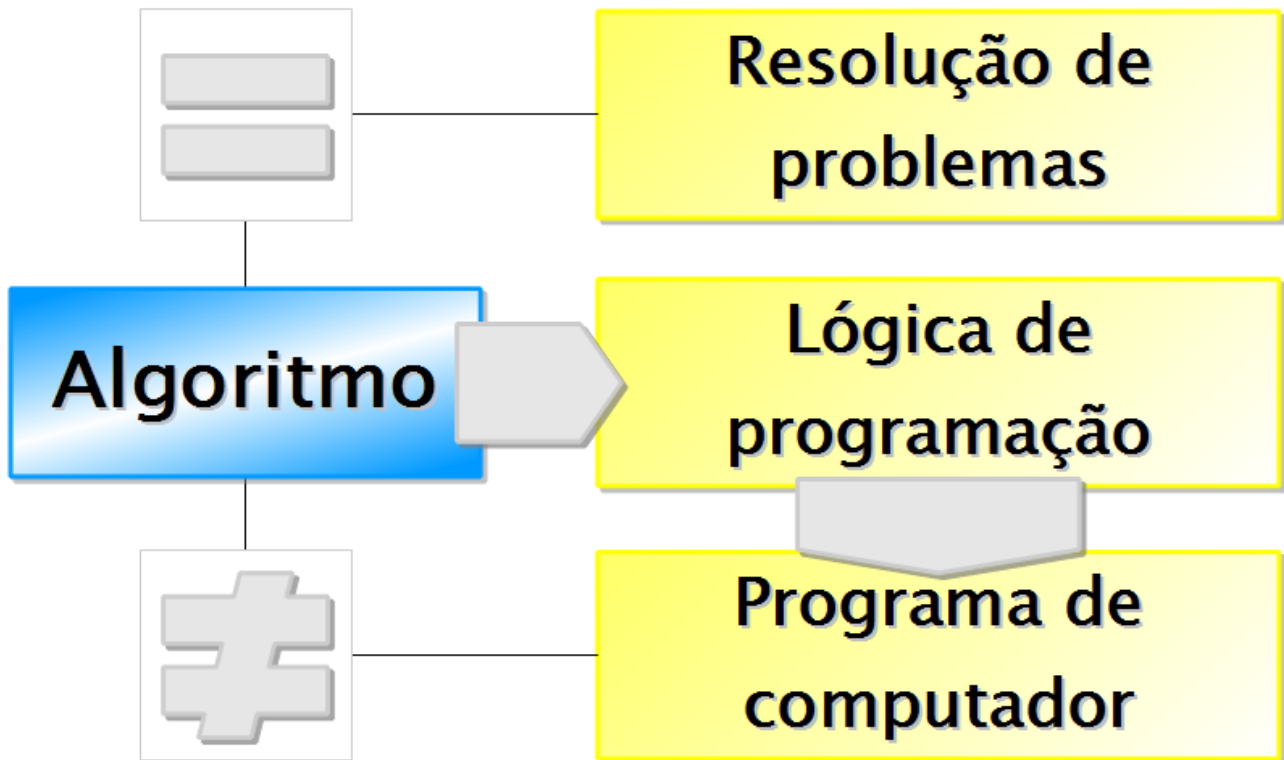


Figura 1.4: Algoritmo, resolução de problemas, lógica de programação e programa de computador.

## 1.2 Algoritmos

Depois do que vimos até agora, o que é realmente um algoritmo? Podemos nos focar nos seguintes conceitos:

- É um **conjunto finito de passos** formalmente definidos para **resolução de um problema ou tarefa**;
- É uma **sequência lógica** que pode ter um conjunto de valores de **entrada** para produzir um conjunto de valores de **saída**; e
- Corresponde a uma descrição de um **padrão de comportamento** expresso em termos de um **conjunto finito de ações**.

Algoritmo é a descrição de uma sequência de passos que deve ser seguida para a realização de uma tarefa (ASCENCIO, 1999).

Um algoritmo consiste simplesmente em uma sequência finita de regras ou instruções que especificam como determinadas operações básicas, executáveis mecanicamente, devem ser combinadas para realização de uma tarefa desejada (CAMARÃO, 2003).

Nota 1.5: Mais conceitos de algoritmo.

Lembram da nota 1.3? Pois é! Por todos os conceitos citados sobre algoritmos, nenhum deles relaciona diretamente à questão de programas de computador. Beleza?

Para que possamos solucionar um problema através de um algoritmo, aquele precisa ser:

- Claro; e
- Bem definido.

O **problema solucionado** por **algoritmo** deve ser:

- **Claro**; e
- **Bem definido**.

Nota 1.6: Tipo de problema solucionado por algoritmo.

Mas por que isso? Porque um dos pontos principais quando criamos um algoritmo é a **finitude de seus passos**. Como podemos definir um conjunto finito de passos se não sabemos aonde queremos realmente chegar?

### 1.3 Características de um algoritmo

Todo algoritmo precisa possuir as seguintes características (figura 1.5):

- **Entrada**: zero ou mais valores de entrada;
- **Saída**: pelo menos um valor é produzido;
- **Clareza ou Definição**: cada passo/instrução/etapa de um algoritmo deve ser claro e não ambíguo;
- **Efetividade**: cada passo/instrução/etapa de um algoritmo deve ser executável; e
- **Finitude**: o algoritmo deve ter um conjunto finito de passos.



**Entrada**



**Clareza ou Definição**



**Efetividade**



**Saída**



**Finitude**

Figura 1.5: Características de um algoritmo.

## 1.4 Fases de um algoritmo

Na construção de um algoritmo, estes passa por três fases fundamentais (figura 1.6):

- **Entrada:** são os dados que serão processados pelo algoritmo;
- **Processamento:** representa os procedimentos necessários para se chegar ao resultado final; e
- **Saída:** são os dados gerados depois do processamento.

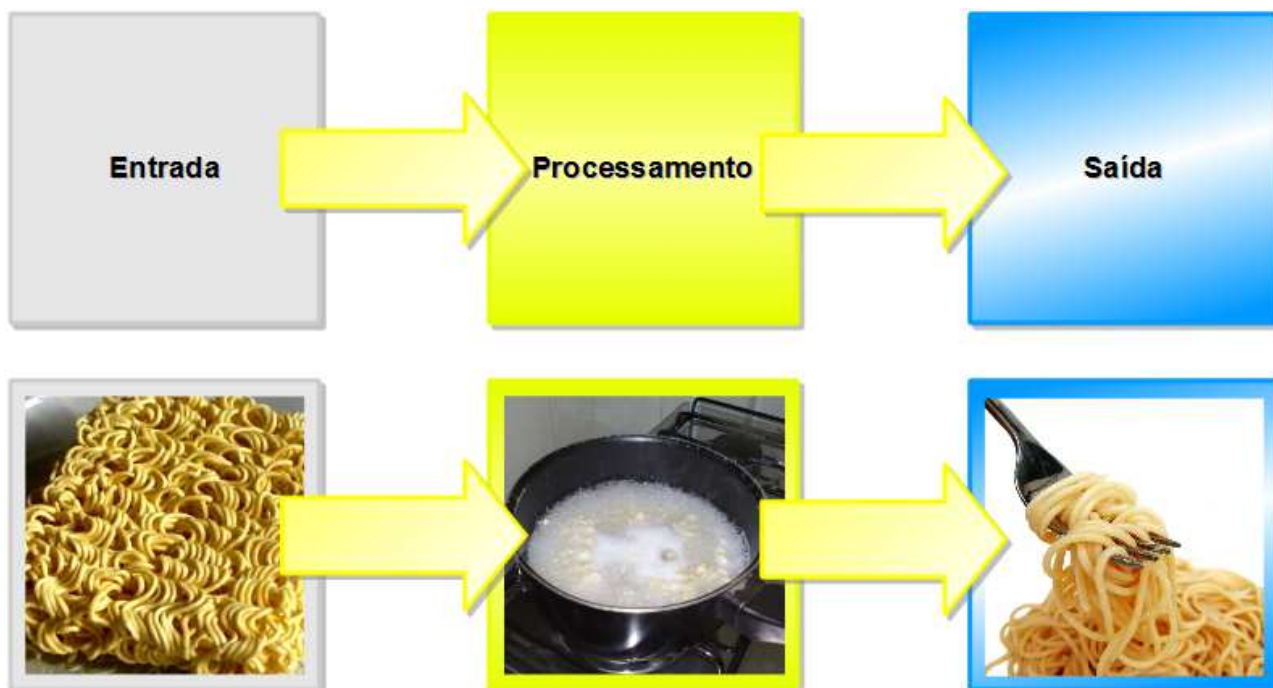


Figura 1.6: Fases de um algoritmo.

Fazendo um paralelo com algo corriqueiro, um macarrãozinho instantâneo seria o dado de entrada para que seja processado (fervura em 3 minutos e, após isso, adicionado o tempero), resultando (saída) em um “delicioso” prato! Deu até fome! ☺

## 1.5 Método para construção de um algoritmo

Um método simples para construção de um algoritmo consiste de:

- Compreendermos o problema a ser resolvido;
  - Por exemplo, o nosso problema pode ser somarmos dois números inteiros;
- Identificarmos e definirmos dos dados de entrada;
  - Para somarmos dois números inteiros, nós precisaremos (adivinhem ☺) um primeiro número inteiro 1 (num1) e outro número inteiro 2 (num2);
- Descrevermos detalhadamente os passos para processar ou transformar os dados de entrada para chegar ao objetivo do problema;
  - Depois da identificação dos dados de entrada, agora é traçarmos como serão os passos do processamento que é somar num1 com num2;
- Identificarmos e definirmos os dados de saída (objetivo do problema)

- Após o processamento, teremos que ter o resultado de saída que o total da soma de num1 com num2;
- Construímos o algoritmo que representa a descrição dos passos;
  - Aqui é praticamente transcrevermos o algoritmo para representar a solução do problema;
- Testarmos o algoritmo para possíveis correções que possam vir a ser necessárias na lógica proposta;
  - Finalmente, depois do algoritmo pronto, realizamos testes para ver se a lógica saiu como deveria ser.

## 1.6 Regras básicas para construção de um algoritmo

Algumas regras precisam ser seguidas para que possamos desenvolver nosso algoritmo:

- Usar somente um verbo por passo/instrução/etapa;
- Escrever de uma forma simples para que possa ser entendido facilmente;
  - Até por pessoas que não trabalham na área;
- Usar frases:
  - Curtas; e
  - Simples;
- Ser objetivo; e
- Procurar usar palavras que não tenham sentido dúbio.

No decorrer do curso, veremos vários exemplos de algoritmos e poderemos ver as regras serem respeitadas. Assim as entenderemos melhor.

## 1.7 Estruturas de controle de fluxo

Um algoritmo, para chegar a um objetivo, compõe-se de uma sequência finita de passos. Para que haja a execução correta desse algoritmo, como organizaremos os seus passos? É aí que entra o conceito de **controle de fluxo**: ele diz respeito a fato de como vamos montar a lógica do algoritmo utilizando certas estruturas. Fazendo uma analogia, a lógica de controle de fluxo seria como um orquestrador e as estruturas utilizadas seriam os orquestrados, tudo isso para que o objetivo almejado seja alcançado pelo resultado final do algoritmo.

Um algoritmo pode ser constituído por três tipos de **estruturas de controle de fluxo**:

- **Estrutura sequencial**: é um bloco de comandos onde cada um deles é executado **passo a passo**, um após o outro;
- **Estrutura condicional** (ou **de seleção** ou **de decisão**): é um bloco de comandos que é executado ou não **dependendo de uma determinada condição** ser verdadeira ou falsa; e
- **Estrutura de repetição** (ou **de iteração** ou **de loop**): é um bloco de comandos executado **repetidas vezes até que uma condição seja alcançada**, encerrando-o e então o fluxo de execução dará continuidade ao restante das ações.

Em qualquer algoritmo, podemos ter as três estruturas. Elas serão vistas com detalhes mais na frente, neste material.

Vamos usar o exemplo 1.1 para entendermos os conceitos vistos agora.

```
Algoritmo preparar macarrão instantâneo;
var
  macarrão: instantâneo;
início
  Coloque o macarrão em uma panela com água;

  Enquanto não der 3 minutos de duração faça:
    Deixe o macarrão fervendo na água;
    Verifique tempo;
  Fim do Enquanto

  Adicione o tempero que vem junto com o macarrão;

  Se o tempero não for o suficiente então:
    Adicione ingredientes a mais como legumes, verduras ou queijo;
  Fim do Se

  Coloque em um prato;
  Sirva;
fim
```

Exemplo 1.1: Algoritmo para preparar macarrão instantâneo.

Pelo exemplo acima, a lógica de controle de fluxo foi feita para prepararmos um macarrão instantâneo e, para isso, utilizamos os três tipos de estruturas vistos nesta seção. Vamos identificar essas estruturas de controle de fluxo usadas no exemplo 1.1?

Estruturas sequenciais:

- Coloque o macarrão em uma panela com água;
- Deixe o macarrão fervendo na água;
- Verifique tempo;
- Adicione o tempero que vem junto com o macarrão;
- Adicione ingredientes a mais como legumes, verduras ou queijo;
- Coloque em um prato;
- Sirva;

Estrutura condicional:

- Se o tempero não for o suficiente então:
  - Adicione ingredientes a mais como legumes, verduras ou queijo;
- Fim do Se

Estrutura de repetição:

- Enquanto não der 3 minutos de duração faça:
  - Deixe o macarrão fervendo na água;



- Verifique tempo;
- Fim do Enquanto

Podemos observar que algumas estruturas sequenciais aparecem tanto na estrutura condicional quanto na estrutura de repetição (nota 1.7).

Em qualquer algoritmo, podemos ter as três estruturas. Elas podem aparecer uma dentro da outra.

Nota 1.7: Estrutura dentro da outra.

No nosso curso, vamos detalhar mais as estruturas que constituem um algoritmo.

Observação 1.1: Detalhes das estruturas que constituem um algoritmo.

## 1.8 Representações de um algoritmo

Existem algumas formas de representação de algoritmos. Dentre elas, destaco:

- **Descrição narrativa:**
  - Os algoritmos são expressos em uma linguagem natural, podendo dar margem a más interpretações, ambiguidades ou imprecisões;
  - Linguagem **mais informal**;
- **Fluxograma:**
  - Os algoritmos são expressos graficamente utilizando-se formas geométricas padronizadas, cada uma com um significado específico;
- **Linguagem algorítmica:**
  - Também chamada de **pseudocódigo, portugol** ou **pseudolinguagem**;
  - Os algoritmos são expressos em uma **linguagem intermediária** entre a **linguagem natural** e uma **linguagem de programação**.

A descrição narrativa não é a melhor opção para representação de algoritmos porque ela pode gerar más interpretações, ambiguidades ou imprecisões. Por exemplo, vejamos a frase:

– Samara está estudando para o Senado.

A partir da frase acima, podemos chegar, pelo menos, a duas conclusões:

- A Samara está estudando para ser uma política e concorrer para ser senadora; ou
- A Samara está estudando para o concurso do Senado.

A segunda conclusão é no nosso contexto, não é, galera? ☺

Bom, com apenas essa descrição bem informal, não temos certeza nenhuma do que ela quer dizer se não estivermos no mesmo contexto de quem a descreveu.

Os fluxogramas nada mais são do que representações gráficas de um pseudocódigo. Visualmente, após um entendimento do que cada forma geométrica significa, fica fácil entender o que propõe o algoritmo representado por um fluxograma, porém dá um trabalhão para mantê-lo porque uma única correção pode requerer uma reorganização de muitas instruções. Alguns dos principais símbolos utilizados em um fluxograma estão listados na tabela 1.1.

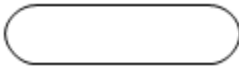



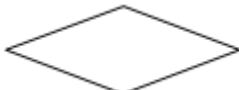

Símbolo	Descrição
	Início ou fim do algoritmo
	Indica o sentido do fluxo de execução do algoritmo. Conecta os objetos gráficos
	Representa a entrada de dados
	Indica cálculos e atribuições de valores (processamento)
	Indica desvios ou tomadas de decisões (Por exemplo: SE isso, ENTÃO aquilo)
	Representa a saída de dados

Tabela 1.1: Símbolos utilizados em um fluxograma.

A linguagem algorítmica, também chamada de pseudocódigo, portugol ou pseudolinguagem, é a mais utilizada por ser mais formal do que a descrição narrativa e mais fácil de manter do que um fluxograma. Além disso, essa linguagem é menos rígida do que uma linguagem de programação, dando assim liberdade ao programador para “rascunhar” seus futuros programas sem ficar engessado na rigidez da linguagem de programação escolhida por ele.

No nosso curso, vamos focar na linguagem algorítmica, pois é representação mais utilizada nos concursos.

Observação 1.2: Escolha da linguagem algorítmica para nosso curso.

## 1.9 Forma geral de um algoritmo (pseudocódigo)

Para pseudocódigo, os algoritmos seguem a forma descrita no exemplo 1.2.

```

Algoritmo <nome do algoritmo>;
  const
    <identificador> = <valor>;
  var
    <identificador>: <tipo>;
  início
    <lista de comandos ou instruções>;
  fim
    
```

Exemplo 1.2: Forma geral de um algoritmo (pseudocódigo).

Na definição de uma constante, igualamos ela a algum valor usando “=”. Já na definição de uma variável, dizemos qual é o seu tipo com “:”.

**Nota 1.8: Diferença nas definições de constantes e variáveis.**

A área de declaração de constantes e variáveis é como se fosse o cabeçalho de um algoritmo onde definimos os dados de entrada. O corpo de um algoritmo, também chamado de **bloco de instruções**, contém o processamento onde os dados de entrada poderão gerar os dados de saída.

O corpo de um algoritmo onde acontece o processamento é chamado de **bloco de instruções**.

**Nota 1.9: Bloco de instruções.**

## 1.10 Itens importantes para um algoritmo

No decorrer da criação de um algoritmo, notamos alguns símbolos que precisam ser explicados para serem entendidos (tabela 1.2).

Item	Símbolo	Descrição
Dois pontos	:	Declara o tipo de uma <b>variável</b> no momento de sua <b>criação</b>
Igualdade	=	Atribui um valor a uma <b>constante</b> no momento de sua <b>criação</b>
Atribuição	:= ou ←	Atribui a uma <b>variável</b> um <b>valor constante</b> , um <b>valor de outra variável</b> ou <b>resultado de alguma operação que retorna valor</b>
Ponto e vírgula	;	Indica a <b>finalização</b> de uma <b>linha de instrução</b>
Chaves	{ }	Traz <b>comentários</b> explicativos do algoritmo e não são considerados no processo de execução
Palavras em negrito		São <b>palavras reservadas</b> que possuem objetivos específicos e não podem ser utilizadas como identificadores para variáveis ou constantes

Tabela 1.2: Itens importantes para um algoritmo.

### 1.10.1 Atribuições importantes

Para o item atribuição, trago alguns exemplos para entendermos seu uso (tabela 1.3).

Atribuição	Exemplo
variável := valor ou variável ← valor	num1 := 15 ou num1 ← 15
variável := variável ou variável ← variável	num1 := num2 ou num1 ← num2

variável := expressão ou variável ← expressão	total := num1 + num2 ou total ← num1 + num2
variável := operação que retorna valor ou variável ← operação que retorna valor	num1 := fatorial(5) ou num1 ← fatorial(5)

Tabela 1.3: Atribuições importantes.

## 1.11 Variáveis e constantes

**Variáveis** e **constantes** são “recipientes” que **armazenam informações** de um **determinado tipo**. Entende-se recipiente como um endereço de memória onde será armazenado um valor.

A diferença entre variáveis e constantes é que as informações contidas nas variáveis podem ser modificadas no decorrer de um algoritmo e as informações relacionadas a constantes são declaradas no início do algoritmo e não podem ser mais modificadas.

Nota 1.10: Diferença entre variáveis e constantes.

Entre algumas semelhanças comuns entre variáveis e constantes, está o fato delas serem identificadas por algum nome.

### 1.11.1 Analogia de variáveis e constantes com garagens de carros

Para podermos entender mais facilmente os conceitos de variáveis e constantes, vamos utilizar uma analogia com garagens de carros (figura 1.7).



Figura 1.7: Variáveis e constantes x garagens de carros.

Imaginem garagens identificadas por alguma numeração. Cada uma recebe apenas um modelo de carro. Por exemplo, há uma garagem apenas para modelos fusca, outra apenas para modelos Civic e assim por diante.

Imaginem também que há dois tipos de garagens. O primeiro tipo possui rotatividade de carros, onde, em determinado momento, uma garagem recebe um carro de um modelo e, em outro momento, pode receber outro carro também do mesmo modelo. O segundo tipo são garagens de pessoas que colecionam carros e que em cada uma há um carro de um modelo, porém, esse carro nunca sairá da sua vaga.

Então, resumindo a analogia, nós temos:

- **Informação:** **carro**;
- **Tipo da informação:** **modelo do carro**;
- **Variáveis:** **garagem numerada** com **rotatividade de carros de mesmo modelo**;
- **Constantes:** **garagem numerada** de um **coleccionador onde ele estaciona um carro de coleção**.

### 1.11.2 Declaração de variáveis e constantes

A declaração de variáveis é feita seguinte forma:

```
var
<identificador1>[, <identificador2>, ...]: <tipo1>;
<identificador3>[, <identificador4>, ...]: <tipo2>;
```

Os elementos entre [] são opcionais.

Já a declaração de constantes difere um pouco:

```
const
<identificador1> = <valor1>;
<identificador2> = <valor2>;
<identificadorN> = <valorN>;
```

Notem que na declaração de variáveis, usamos “:” seguido do tipo (tabela 1.3). Na declaração de constantes, usamos “=” seguido de um valor e o que define o tipo de uma constante é o tipo desse valor atribuído a ela.

O que define o tipo de uma constante é o tipo do valor atribuído a ela.

Nota 1.11: Tipo de constantes.

### 1.11.3 Regras para nomes de variáveis e constantes

Para identificarmos variáveis e constantes, precisamos seguir algumas regras:

- Não podem:

- Ter nomes de palavras reservadas (comandos de uma linguagem de programação específica);
- Possuir espaços em branco;
- Devem iniciar com:
  - Letra; e
  - Sublinhado (\_);
- Podem ter como demais caracteres:
  - Letras;
  - Números; e
  - Sublinhado.

Destaco dois pontos importantes:

- Para identificação de constantes, por convenção, usa-se letras maiúsculas; e
- Para algoritmos, a escolha de letras maiúsculas ou minúsculas para nomes de variáveis é indiferente.

**Nota 1.12: Pontos importantes para identificação de variáveis e constantes.**

```
var
    nota1, nota2, nota3: inteiro;
    peso: real;
    _nome123: caractere;
const
    PI = 3,14;
```

**Exemplo 1.3: Nomes de variáveis e constantes.**

## 1.12 Algoritmo 1 – Encontrar o maior número de dois inteiros

Vamos colocar em prática o que vimos até agora. Afinal, na prática é que realmente aprendemos. ☺

Para nosso primeiro problema, temos que fazer um algoritmo que possa receber dois números inteiros para encontrar o maior deles. Veremos a solução em quatro formas diferentes:

- Em português (nosso principal objetivo);
- Em fluxograma (apenas para vermos como seria um);
- Em português no VisualAlg; e
- Na linguagem Java (uma das linguagens mais cobradas em concursos).

No nosso curso, os demais algoritmos para os problemas propostos serão vistos em português, português no VisualAlg e na linguagem Java.

**Observação 1.3: Representações de algoritmos nos problemas propostos.**

```
Algoritmo maiorDeDoisNumerosInteiros;  
var  
  num1, num2, maior: inteiro;  
início  
  escreva("Digite o valor do número 1: ");  
  leia(num1); {recebe o primeiro valor de entrada}  
  escreva("Digite o valor do número 2: ");  
  leia(num2); {recebe o segundo valor de entrada}  
  
  se (num1 > num2) então  
    maior := num1; {define para a variável maior o valor da variável num1}  
  senão  
    maior := num2; {define para a variável maior o valor da variável num2}  
  fim do se  
  
  escreva("O maior número é ", maior);  
fim
```

Exemplo 1.4: Algoritmo do problema 1 em português.

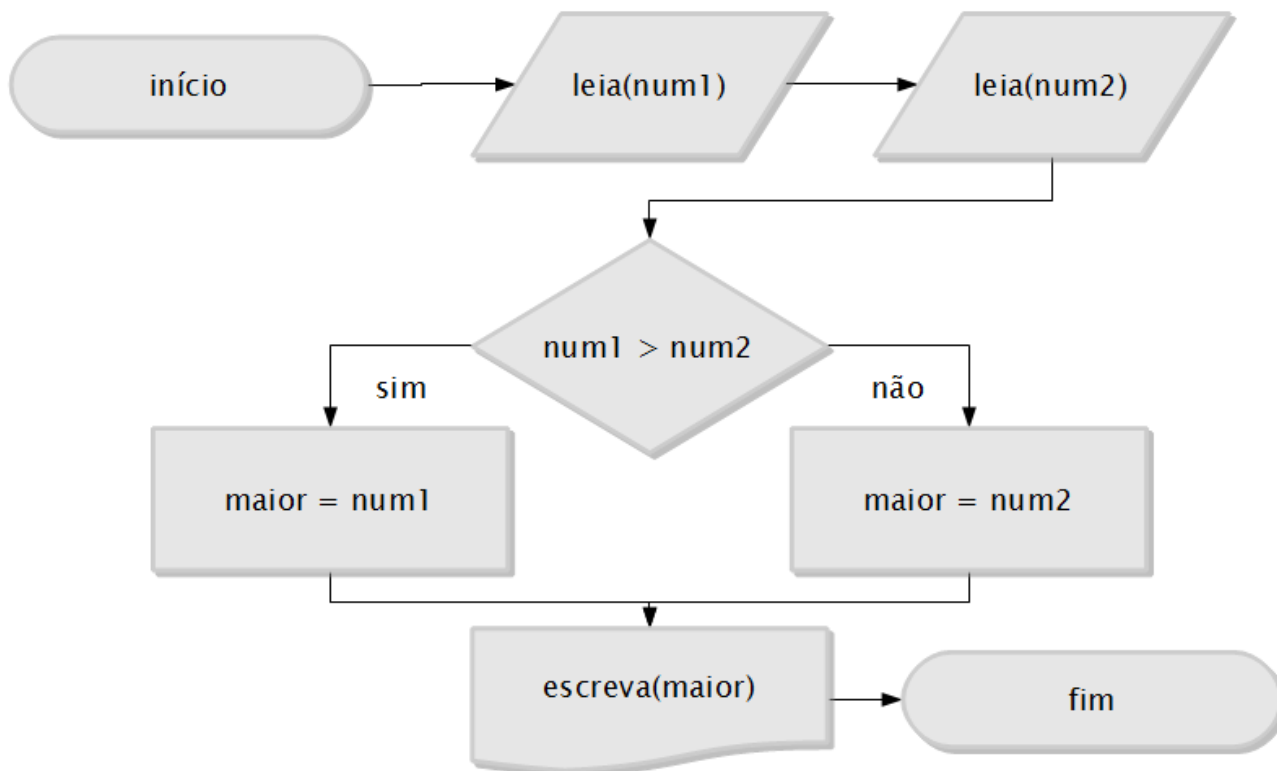


Figura 1.7: Fluxograma do problema 1.

```
algoritmo "maiorDeDoisNumerosInteiros"  
  
var  
    num1 : inteiro  
    num2 : inteiro  
    maior : inteiro  
  
inicio  
    escreva ("Digite o valor do número 1: ")  
    leia (num1)  
    escreva ("Digite o valor da número 2: ")  
    leia (num2)  
  
    se (num1 > num2) entao  
        maior <- num1  
    senao  
        maior <- num2  
    fimse  
  
    escreva ("O maior número é: ", maior)  
finalgoritmo
```

Exemplo 1.5: Algoritmo do problema 1 em portugol no VisualAlg.

```
public class MaiorDeDoisNumerosInteiros {  
  
    public static void main(String[] args) {  
        int num1;  
        int num2;  
        int maior;  
        Scanner entrada = new Scanner(System.in);  
  
        System.out.print("Digite o valor do número 1: ");  
        num1 = entrada.nextInt();  
  
        System.out.print("Digite o valor do número 2: ");  
        num2 = entrada.nextInt();  
  
        entrada.close();  
    }  
}
```



```
    if (num1 > num2)
        maior = num1;
    else
        maior = num2;

    System.out.println("O maior número é " + maior);
}
}
```

Exemplo 1.6: Algoritmo do problema 1 na linguagem Java.

## 1.13 Entrada e saída de dados

### 1.13.1 Entrada

Muitas vezes, um algoritmo, para que possa trabalhar, precisa de dados de entrada informados pelo usuário. Digo muitas vezes porque nem sempre é necessário ter esse tipo de dados, porém, pelo menos um dado de saída é gerado (vejam novamente a seção 1.3 Características de um algoritmo).

A entrada de dados é feita pelo comando [leia](#). Vejam o exemplo 1.7.

### 1.13.2 Algoritmo 2 – Ler um número inteiro

```
Algoritmo lerNumeroInteiro;
var
    num: inteiro;
início
    escreva("Digite um número: ");
    leia(num);
    escreva("O número digitado foi ", num);
fim
```

Exemplo 1.7: Algoritmo que lê um número inteiro.

```
algoritmo "lerNumeroInteiro"
var
    num : inteiro
início
    escreva ("Digite um número: ")
    leia (num)
    escreva ("O maior número é: ", maior)
```

[fimalgoritmo](#)**Exemplo 1.8: Algoritmo que lê um número inteiro no VisualAlg.**

```
public class LerNumeroInteiro {  
  
    public static void main(String[] args) {  
        int num;  
        Scanner entrada = new Scanner(System.in);  
  
        System.out.print("Digite um número: ");  
        num = entrada.nextInt();  
  
        entrada.close();  
  
        System.out.println("O número digitado foi " + num);  
    }  
}
```

**Exemplo 1.9: Algoritmo que lê um número inteiro na linguagem Java.**

Necessariamente, para a leitura de entrada de dados, precisamos usar variáveis. Vocês lembram da nossa analogia da sessão 1.11.1 (Analogia de variáveis e constantes com garagens de carros)? Pois é! Quando entramos com um dado (veículo de um modelo), indicamos qual variável (garagem) vai recebê-lo.

Necessariamente, para a leitura de entrada de dados, precisamos usar variáveis.

Nota 1.13: Necessidade de uso de variáveis para entrada de dados.

### 1.13.3 Saída

Para mostrarmos o resultado do processamento do algoritmo, usamos o comando **escreva**. O comando recebe um texto e podemos intercalá-lo com variáveis e constantes, usando vírgulas (exemplo 1.10).

### 1.13.4 Algoritmo 3 - Ler um número inteiro (modificado)

```
Algoritmo lerNumeroInteiro;  
  
var  
    num: inteiro;  
  
início  
    escreva("Digite um número: ");
```

```
leia(num);
escreva("O número digitado foi ", num);
escreva("O número ", num, " foi digitado");
fim
```

Exemplo 1.10: Algoritmo que lê um número inteiro (modificação do exemplo 1.9).

## 1.14 Teste de mesa ou teste chinês

Uma forma para testarmos nossos algoritmos é utilizando uma técnica chamada **teste de mesa** ou **teste chinês**. Essa técnica consiste de acompanharmos passo a passo um algoritmo, de forma a procurarmos falhas na lógica utilizada para resolver um determinado problema.

Um algoritmo apenas é correto se produzir o resultado esperado para qualquer entrada informada. Pode ocorrer que para certas entradas o resultado seja como o esperado e para outras entradas, não seja. A aplicação do teste de mesa pode mostrar onde podemos consertar a lógica.

Um algoritmo apenas é correto se produzir o resultado esperado para qualquer entrada informada.

Nota 1.14: Algoritmo correto.

### 1.14.1 Algoritmo 4 – Calcular salário a receber

Vamos utilizar exemplo 1.11 para colocarmos em prática a técnica. Para isso, vamos numerar as linhas do algoritmo. O nosso problema será calcular o salário a receber seguindo os seguintes itens:

- Informar o salário-base;
- Haverá uma gratificação que é 5% do valor do salário-base;
- Haverá um imposto que é 3% do valor do salário-base; e
- O salário a receber é a soma do salário-base com a gratificação descontado o imposto.

```
Algoritmo calcularSalarioReceber;
var
    salarioBase, gratificacao, imposto, salarioReceber: real;

início
1  escreva("Informe o salário-base: ");
2  leia(salarioBase);
3  gratificacao := salarioBase * 5 / 100;
4  imposto := salarioBase * 3 / 100;
5  salarioReceber := salarioBase + gratificacao - imposto;
6  escreva("O salário a receber é ", salarioReceber);
fim
```

Exemplo 1.11: Algoritmo que calcula salário a receber.

Para facilitar nossa inspeção do algoritmo linha à linha, utilizamos uma tabela onde preenchemos com os valores de cada variável em um determinado passo do algoritmo. Para nosso teste, o salário-base será de R\$ 1.000,00.

Linha	salarioBase	gratificacao	imposto	salarioReceber
1	-	-	-	-
2	1000	-	-	-
3	1000	50	-	-
4	1000	50	30	-
5	1000	50	30	1020
6	1000	50	30	1020

Tabela 1.3: Teste de mesa do exemplo 1.9.

Para um algoritmo, podemos fazer vários testes de mesa, informando várias entradas diferentes para saber se o algoritmo trabalha de forma consistente para qualquer entrada.

```

algoritmo "calcularSalarioReceber"

var
    salarioBase, gratificacao, imposto, salarioReceber : real
inicio
    escreva ("Informe o salário-base: ")
    leia (salarioBase)
    gratificacao := salarioBase * 5 / 100
    imposto := salarioBase * 3 / 100
    salarioReceber := salarioBase + gratificacao - imposto
    escreva ("O salário a receber é ", salarioReceber)
fimalgoritmo
    
```

Exemplo 1.12: Algoritmo que calcula salário a receber no VisualAlg.

```

public class CalcularSalarioReceber {

    public static void main(String[] args) {
        int salarioBase, gratificacao, imposto, salarioReceber;
    }
}
    
```

```
Scanner entrada = new Scanner(System.in);
DecimalFormat formato = new DecimalFormat("0.00");

System.out.print("Informe o salário-base: ");
salarioBase = entrada.nextInt();

entrada.close();

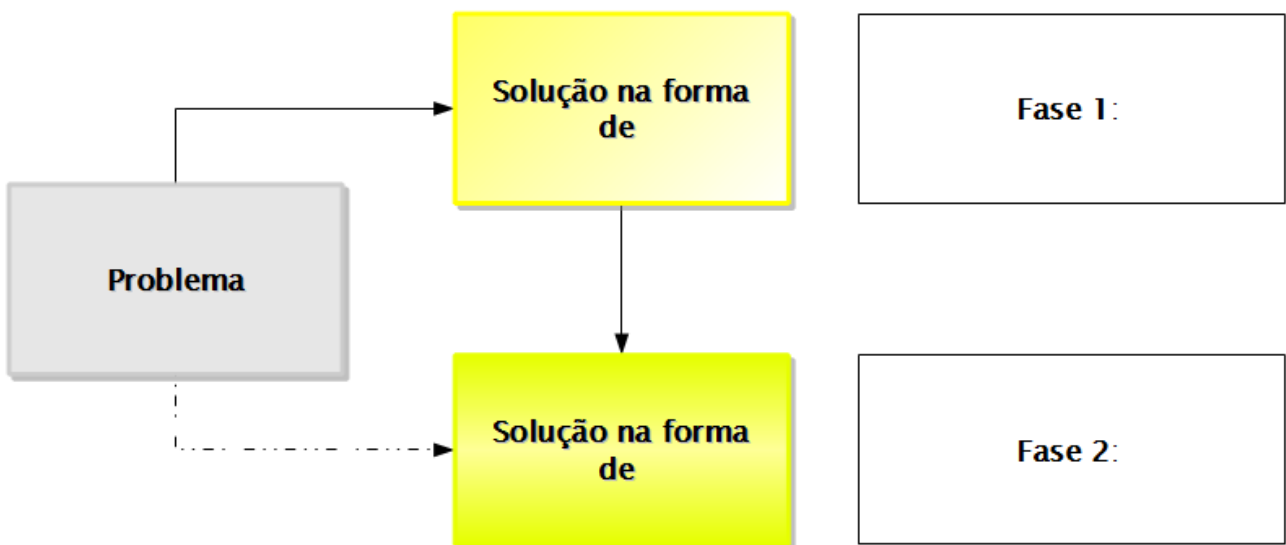
gratificacao = salarioBase * 5 / 100;
imposto = salarioBase * 3 / 100;
salarioReceber = salarioBase + gratificacao - imposto;

System.out.println("O salário a receber é " +
formato.format(salarioReceber));
}
}
```

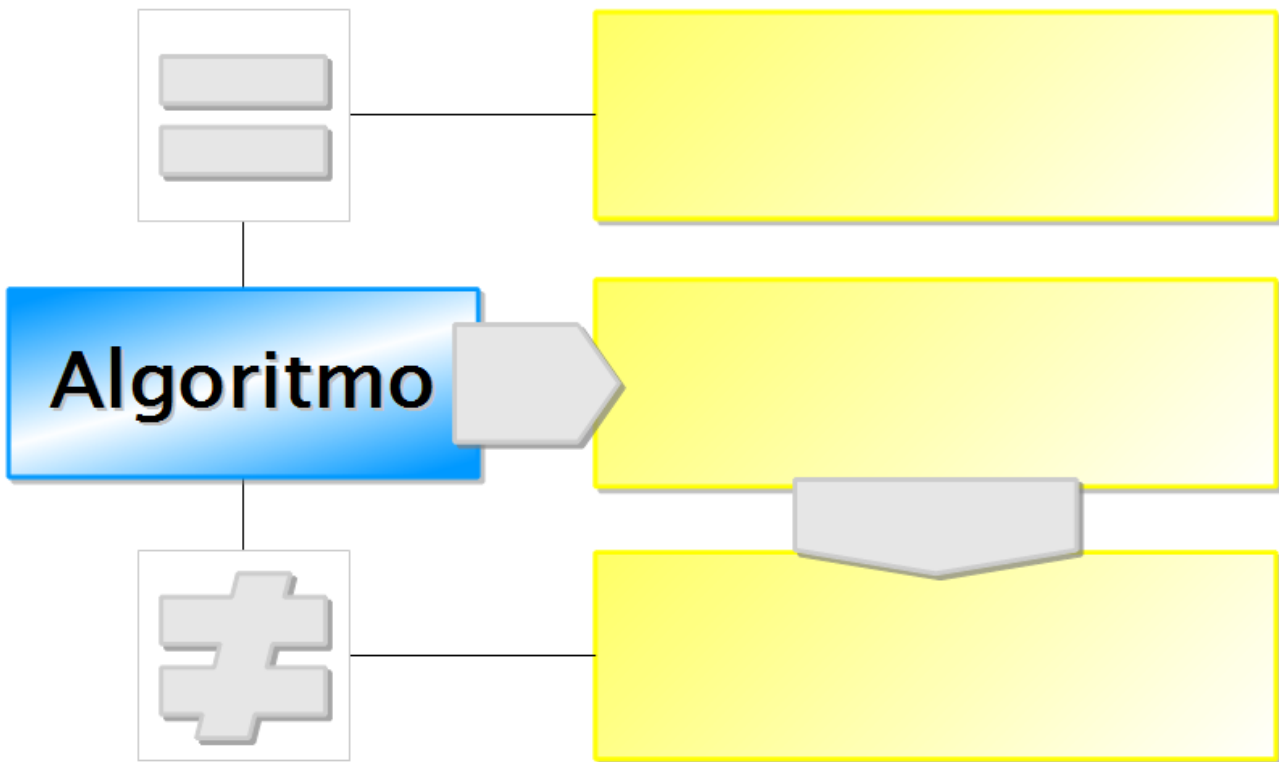
Exemplo 1.13: Algoritmo que calcula salário a receber na linguagem Java.

### 1.15 Questões do capítulo

- 1 Entre outras habilidades, o que exige a construção de programas de computador?
- 2 Quando falamos de capacidade de resolução de problemas, estamos falando de algoritmos (CERTO/ERRADO).
- 3 Quando se fala em algoritmo, podemos dizer que é um programa de computador feito para resolver computacionalmente um problema específico (CERTO/ERRADO).
- 4 Preencha a imagem de acordo com o que foi visto no capítulo:



5 Preencha a imagem de acordo com o que foi visto no capítulo:



6 Algoritmo é um conjunto infinito de passos formalmente definidos para resolução de um problema ou tarefa (CERTO/ERRADO).

7 O problema solucionado por algoritmo deve ser claro e bem definido (CERTO/ERRADO).

8 Quais são as características de um algoritmo?




9 Quais são as fases de um algoritmo?

10 Em um algoritmo, podemos usar mais de um verbo por passo/instrução/etapa (CERTO/ERRADO).

11 Quais são as estruturas de controle de fluxo que constituem um algoritmo?

12 Quais são as representações de um algoritmo?

13 Preencha a tabela abaixo:

Símbolo	Descrição
	
	
	

14 Qual é a forma geral de um algoritmo?

15 O corpo de um algoritmo, também chamado de [...], contém o processamento onde os dados de entrada poderão gerar os dados de saída.

16 Preencha a tabela abaixo:

Item	Símbolo	Descrição
Dois pontos	:	
Igualdade	=	
Atribuição	:= ou ←	
Ponto e vírgula	;	
Chaves	{ }	
Palavras em negrito		

17 Para declarar variáveis, fazemos da seguinte forma: <identificador1>[, <identificador2>, ...] = <tipo1>;. Para constantes, usamos: <identificador1>: <valor1>; (CERTO/ERRADO).

18 O que define o tipo de uma constante?

19 Nomes de variáveis e constantes podem começar com letras ou sublinhado (CERTO/ERRADO).

20 Necessariamente, para a leitura de entrada de dados, precisamos usar variáveis (CERTO/ERRADO).

21 O que é um algoritmo correto?

## 1.16 Respostas das questões do capítulo

1 Entre outras habilidades, o que exige a construção de programas de computador?

A **construção de programas de computador** exige, entre outras habilidades, a **capacidade de resolver problemas** através da **identificação de um conjunto ordenado e finito de etapas e/ou instruções** que levam a sua resolução.

2 Quando falamos de capacidade de resolução de problemas, estamos falando de algoritmos (CERTO/ERRADO).

Gabarito: CERTO.

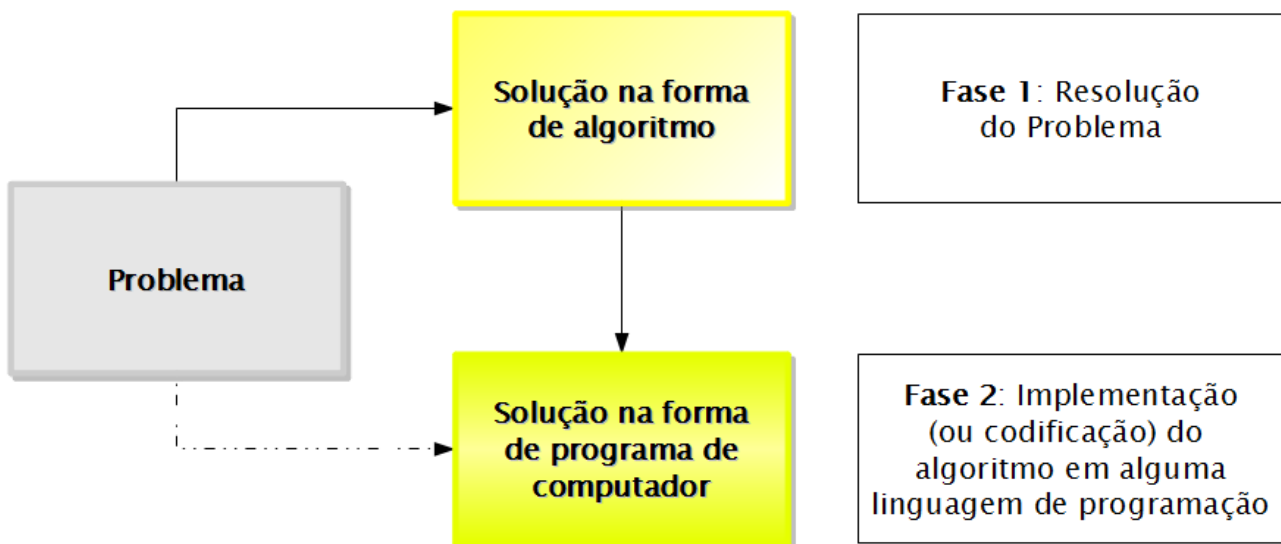
Para construirmos nossos programas de computador, precisamos entender a importância do conceito de **algoritmos**. Quero frisar que quando falamos de capacidade de resolução de problemas, estamos falando de algoritmos.

3 Quando se fala em algoritmo, podemos dizer que é um programa de computador feito para resolver computacionalmente um problema específico (CERTO/ERRADO).

Gabarito: ERRADO.

O **conceito de algoritmo não é de exclusividade** da **área da computação**. Um algoritmo também **não representa**, necessariamente, um **programa de computador** e sim os passos necessários para realizar uma tarefa ou solucionar um problema, seja de que área for.

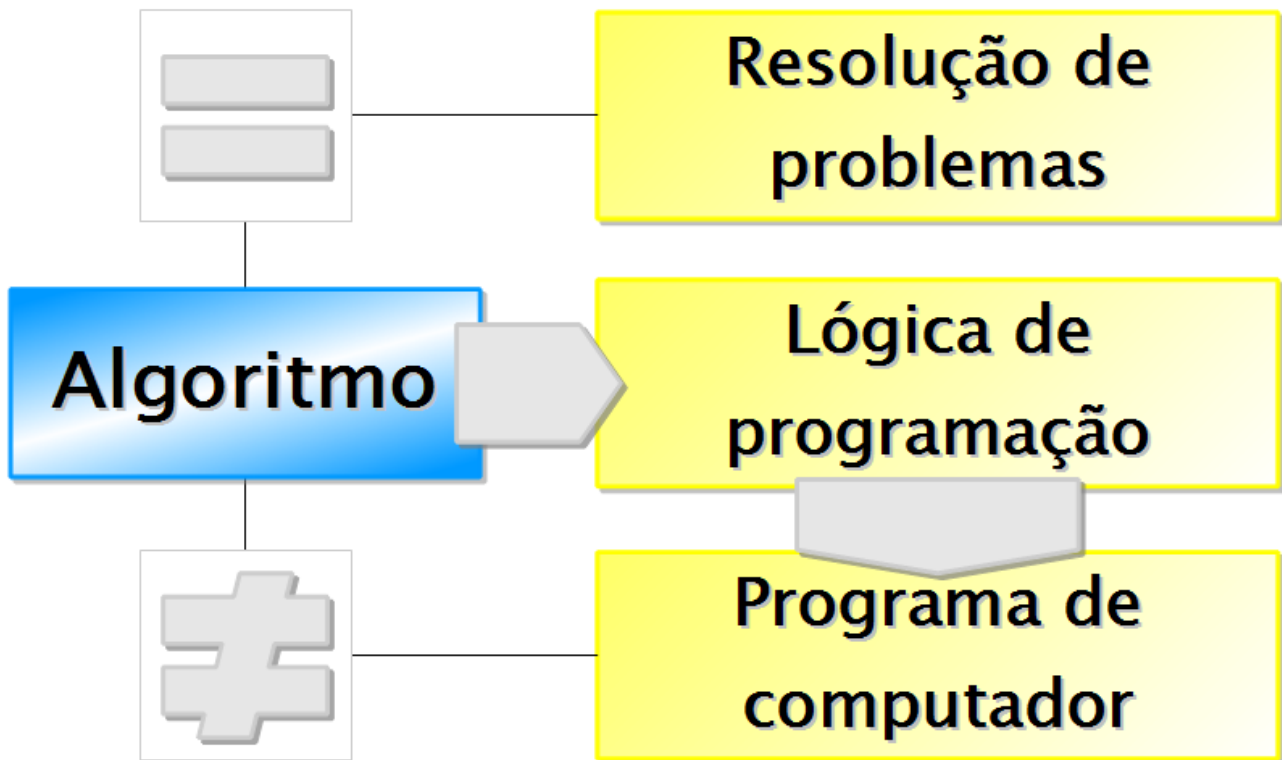
4 Preencha a imagem de acordo com o que foi visto no capítulo:



Quando temos um problema para ser traduzido em programa de computador, um passo importante é resolver o problema na forma de algoritmo. A partir disso, já podemos implementar o algoritmo em alguma linguagem de programação.



5 Preencha a imagem de acordo com o que foi visto no capítulo:



Revisão dos conceitos:

- Quando falamos de capacidade de resolução de problemas, estamos falando de algoritmos;
- A lógica nos dará o poder de raciocínio de como resolver uma tarefa ou problema;
- No caso de construção de um programa de computador, a lógica de programação consiste em aprender a pensar na mesma sequência de execução desse tipo de programa; e
- Um algoritmo não representa, necessariamente, um programa de computador, e sim os passos necessários para realizar uma tarefa ou solucionar um problema.

6 Algoritmo é um conjunto infinito de passos formalmente definidos para resolução de um problema ou tarefa (CERTO/ERRADO).

Gabarito: **ERRADO**.

É um **conjunto finito de passos** formalmente definidos para **resolução de um problema ou tarefa**.

7 O problema solucionado por algoritmo deve ser claro e bem definido (CERTO/ERRADO).

Gabarito: **CERTO**.

Um dos pontos principais quando criamos um algoritmo é a **finitude de seus passos**. Como podemos definir um conjunto finito de passos se não sabemos aonde queremos realmente chegar?

8 Quais são as características de um algoritmo?

- **Entrada:** zero ou mais valores de entrada;
- **Saída:** pelo menos um valor é produzido;
- **Clareza ou Definição:** cada passo/instrução/etapa de um algoritmo deve ser claro e não ambíguo;
- **Efetividade:** cada passo/instrução/etapa de um algoritmo deve ser executável; e
- **Finitude:** o algoritmo deve ter um conjunto finito de passos.



**Entrada**



**Efetividade**



**Saída**



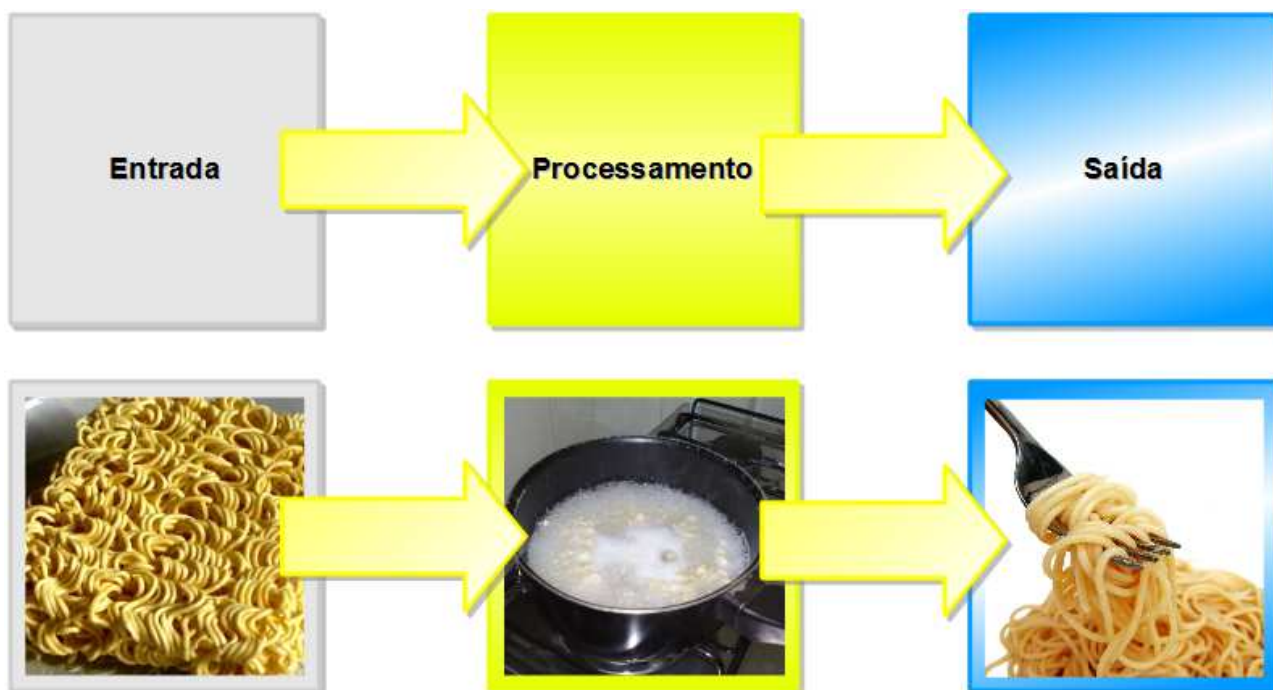
**Clareza ou Definição**



**Finitude**

9 Quais são as fases de um algoritmo?

- **Entrada:** são os dados que serão processados pelo algoritmo;
- **Processamento:** representa os procedimentos necessários para se chegar ao resultado final; e
- **Saída:** são os dados gerados depois do processamento.



10 Em um algoritmo, podemos usar mais de um verbo por passo/instrução/etapa (CERTO/ERRADO).

Gabarito: **ERRADO**.

Algumas regras precisam ser seguidas para que possamos desenvolver nosso algoritmo:

- Usar somente um verbo por passo/instrução/etapa;
- Escrever de uma forma simples para que possa ser entendido facilmente;
  - Até por pessoas que não trabalham na área;
- Usar frases:
  - Curtas; e
  - Simples;
- Ser objetivo; e
- Procurar usar palavras que não tenham sentido dúbio.

Como visto, devemos usar somente um verbo por passo/instrução/etapa.

11 Quais são as estruturas de controle de fluxo que constituem um algoritmo?





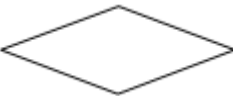

- **Estrutura sequencial**: é um bloco de comandos onde cada um deles é executado **passo a passo**, um após o outro;
- **Estrutura condicional** (ou **de seleção** ou **de decisão**): é um bloco de comandos que é executado ou não **dependendo de uma determinada condição** ser verdadeira ou falsa; e
- **Estrutura de repetição** (ou **de iteração** ou **de loop**): é um bloco de comandos executado **repetidas vezes até que uma condição seja alcançada**, encerrando-o e então o fluxo de execução dará continuidade ao restante das ações.

12 Quais são as representações de um algoritmo?

- **Descrição narrativa**:

- Os algoritmos são expressos em uma linguagem natural, podendo dar margem a más interpretações, ambiguidades ou imprecisões;
- Linguagem **mais informal**;
- **Fluxograma:**
  - Os algoritmos são expressos graficamente utilizando-se formas geométricas padronizadas, cada uma com um significado específico;
- **Linguagem algorítmica:**
  - Também chamada de **pseudocódigo**, **portugol** ou **pseudolinguagem**;
  - Os algoritmos são expressos em uma **linguagem intermediária** entre a **linguagem natural** e uma **linguagem de programação**.

13 Preencha a tabela abaixo:

Símbolo	Descrição
	Início ou fim do algoritmo
	Indica o sentido do fluxo de execução do algoritmo. Conecta os objetos gráficos
	Representa a entrada de dados
	Indica cálculos e atribuições de valores (processamento)
	Indica desvios ou tomadas de decisões (Por exemplo: SE isso, ENTÃO aquilo)
	Representa a saída de dados

14 Qual é a forma geral de um algoritmo?

```

Algoritmo <nome do algoritmo>;
  const
    <identificador> = <valor>;
  var
    <identificador>: <tipo>;
  início
    <lista de comandos ou instruções>;
  fim
    
```

15 O corpo de um algoritmo, também chamado de **bloco de instruções**, contém o processamento onde os dados de entrada poderão gerar os dados de saída.

16 Preencha a tabela abaixo:

Item	Símbolo	Descrição
Dois pontos	:	Declara o tipo de uma <b>variável</b> no momento de sua <b>criação</b>
Igualdade	=	Atribui um valor a uma <b>constante</b> no momento de sua <b>criação</b>
Atribuição	:= ou ←	Atribui a uma <b>variável</b> um <b>valor constante</b> , um <b>valor de outra variável</b> ou <b>resultado de alguma operação que retorna valor</b>
Ponto e vírgula	;	Indica a <b>finalização</b> de uma <b>linha de instrução</b>
Chaves	{ }	Traz <b>comentários</b> explicativos do algoritmo e não são considerados no processo de execução
Palavras em negrito		São <b>palavras reservadas</b> que possuem objetivos específicos e não podem ser utilizadas como identificadores para variáveis ou constantes

17 Para declarar variáveis, fazemos da seguinte forma: *<identificador1>*[, *<identificador2>*, ...] = *<tipo1>*;. Para constantes, usamos: *<identificador1>*: *<valor1>*; (CERTO/ERRADO).

Gabarito: **ERRADO**.

Na definição de uma constante, igualamos ela a algum valor usando “=”. Já na definição de uma variável, dizemos qual é o seu tipo com “:”. A questão inverteu os símbolos.

18 O que define o tipo de uma constante?

O que define o tipo de uma constante é o tipo do valor atribuído a ela.

19 Nomes de variáveis e constantes podem começar com letras ou sublinhado (CERTO/ERRADO).

Gabarito: **CERTO**.

```

var
    nota1, nota2, nota3: inteiro;
    peso: real;
    _nome123: caractere;

const
    PI = 3,14;
    
```

20 Necessariamente, para a leitura de entrada de dados, precisamos usar variáveis (CERTO/ERRADO).

Gabarito: **CERTO**.

## 21 O que é um algoritmo correto?

Um algoritmo apenas é correto se produzir o resultado esperado para qualquer entrada informada.

## 1.17 Questões de concursos

### IESES 2010 CRM/DF – Assistente de Tecnologia da Informação – Adaptada

Em relação à lógica de programação, considere os pseudocódigos:

- I. Um algoritmo corresponde a uma sequência ordenada, e sem ambiguidade, de ações que levam à solução de um problema e, quando codificado em uma linguagem de programação, corresponde a um programa de computador.

### CESPE 2010 ABIN – Cargo 17: Oficial Técnico de Inteligência – Área de Suporte a Rede de Dados

Julgue os itens seguintes, relativos a programação básica.

79 As estruturas de controle sequenciais, de seleção (ou de decisão) e de repetição (ou de iteração ou loop) são unidades básicas na escrita de algoritmos. Todas essas estruturas possuem condições a serem testadas; algumas realizam atribuição de variáveis, mas somente uma pode inicializar variáveis.

### FEPESE 2010 SEFAZ/SC – Auditor Fiscal da Receita Estadual – Parte III – Tecnologia da Informação

Assinale a alternativa correta a respeito das variáveis e constantes, utilizadas em diversas linguagens de programação.

- A) O número de constantes deve ser menor ou igual ao número de variáveis em um programa.
- B) O número de constantes deve ser menor ou igual ao número de procedimentos em um programa.
- C) O número de constantes deve ser igual ao número de variáveis em um programa.
- D) O número de constantes independe da quantidade de variáveis em um programa.
- E) O número de constantes deve ser igual ao número de procedimentos em um programa.

### FCC 2008 Metrô/SP – Analista Trainee – Ciências da Computação

39 Em relação à lógica de programação, considere os pseudocódigos:

#### Algoritmo Alg1

```
salBase, salReceber, grat, imp: real
Inicio
Leia(salBase)
Grat ← salBase * 5/100
SalReceber ← salBase +grat – Imp
Imp ← SalReceber * 7/100
SalReceber ← SalReceber – imp
Escreva (salReceber)
Fim
```

#### Algoritmo 2

```
salBase, salReceber, Imp: real
Inicio
Leia(salBase)
SalReceber ← salBase+(salBase * 5/100 )
Imp ← SalReceber * 7/100
SalReceber ← SalReceber – imp
Escreva (salReceber)
Fim
```

É correto afirmar:

- A) Somente Alg1 tem consistência em sua representação e chega a um resultado.
- B) Ambos os algoritmos abordam o mesmo problema e chegam ao mesmo resultado.
- C) Somente Alg2 tem consistência em sua representação e chega a um resultado.
- D) O resultado da solução apresentada por Alg2 é maior do que a de Alg1.
- E) O resultado da solução apresentada por Alg2 é menor do que a de Alg1.

#### ESAF 2006 SUSEP – Analista Técnico – Tecnologia da Informação – Adaptada

47 Acerca dos conceitos fundamentais de lógica de programação e algoritmos, é incorreto afirmar que

- B) compreender o problema, selecionar um método de solução, descrever a solução passo a passo, validar o algoritmo, programá-lo e testá-lo, nesta seqüência, é uma proposta viável para analisar um problema.

#### CESPE 2011 STM – Analista de Sistemas

Com relação a algoritmos e lógica de programação, julgue os itens a seguir.

80 Nas estruturas de controle, tais como as estruturas de seleção simples, compostas ou encadeadas, é necessário verificar as condições para a realização de uma instrução ou seqüência de instruções.

#### FGV 2009 MEC – Arquiteto de Sistemas

41 Analise o trecho de algoritmo a seguir, em pseudocódigo:

```
atribuir 'BRASIL-COPA2014' a STR;
atribuir 35 a GAMA;
atribuir 0 a BETA;
repetir
    se (resto da divisão de GAMA por 2 igual a 1)
        então imprimir (STR);
        atribuir GAMA-7 a GAMA;
        se GAMA = 7 então atribuir 1 a BETA;
até que BETA = 1;
```

Após a execução, a variável STR será impressa uma quantidade de vezes igual a:

- A) 0
- B) 1
- C) 2
- D) 3
- E) 4

CESGRANRIO 2009 Cada da Moeda –Analista de Nível Superior – Negócios em TI

18 Analise o pseudocódigo a seguir.

```
1.var n: inteiro
2.escreva ("Digite um número inteiro:")
3.leia(n)
4.n<-n+5
5.escreva(n)
```

Considerando-se que o programa recebeu, como entrada, o valor 10, qual o resultado na tela da execução?

- A) 0
- B) 5
- C) 10
- D) 15
- E) 20

CESPE 2009 CEHAP/PB – Cargo 16: Programador

25 Considere o trecho de código a seguir.



```
INICIO
INTEIRO J, X;
J=1;
X=2;
ENQUANTO J<10 FAÇA
  X = X + 1;
  J = J + 2;
FIM ENQUANTO;
IMPRIMA X;
IMPRIMA J;
```

Ao final da execução do trecho de código acima, os valores de

- A) 7 e 11
- B) 6 e 12
- C) 8 e 11
- D) 9 e 12

**38** Considere o trecho de código a seguir.

```
INICIO
INTEIRO A, B, C, D;
A=10;
B=5;
C=0;
D=0;
ENQUANTO A>10 FAÇA
  C= A + B;
  D= B - A;
  A= A-1;
FIM ENQUANTO;

IMPRIMA A;
IMPRIMA B;
IMPRIMA C;
IMPRIMA D;
```

Ao final da execução do trecho de código acima, os valores de A, B, C e D, são iguais, respectivamente, a

- A) 9, 5, 15 e 15.
- B) 10, 5, 15 e 15.
- C) 10, 5, 0 e 0.
- D) 10, 5, 15 e 5.

## 1.18 Gabarito comentado das questões de concursos

IESES 2010 CRM/DF – Assistente de Tecnologia da Informação – Adaptada

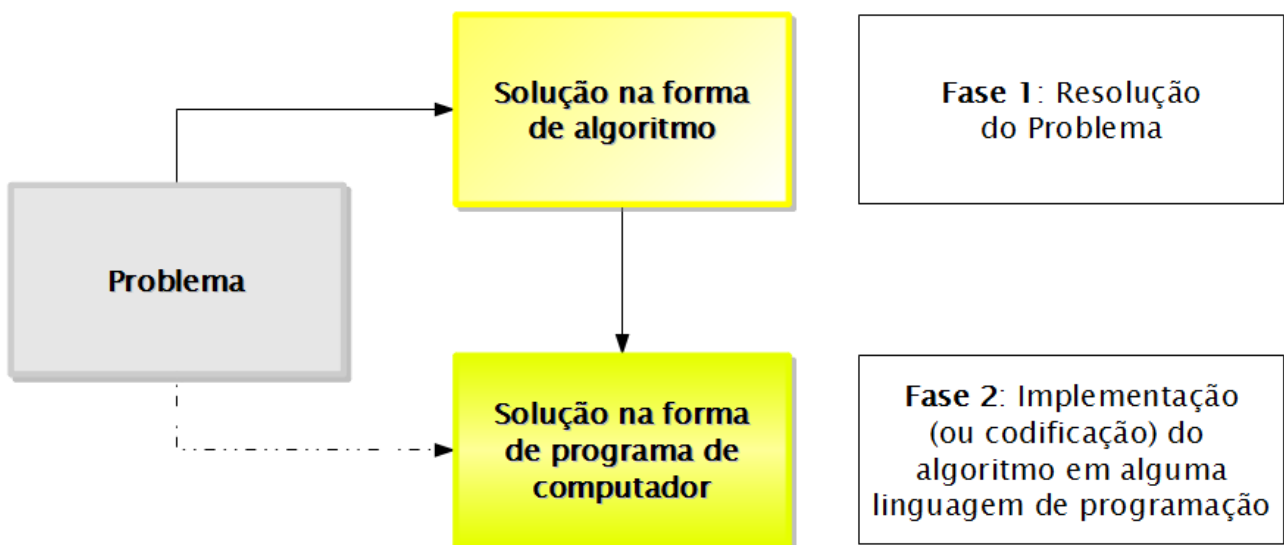
Em relação à lógica de programação, considere os pseudocódigos:

- I. Um algoritmo corresponde a uma sequência ordenada, e sem ambiguidade, de ações que levam à solução de um problema e, quando codificado em uma linguagem de programação, corresponde a um programa de computador.

#### Comentário da questão

Praticamente, o item acima fez um resumo do capítulo. Relembrando, a **construção de programas de computador** exige, entre outras habilidades, a **capacidade de resolver problemas** através da **identificação de um conjunto ordenado e finito de etapas e/ou instruções** que levam a sua resolução.

Também podemos utilizar a imagem abaixo para visualizar o que o item citou.



Gabarito: **CERTO**.

CESPE 2010 ABIN – Cargo 17: Oficial Técnico de Inteligência – Área de Suporte a Rede de Dados

Julgue os itens seguintes, relativos a programação básica.

**79** As estruturas de controle sequenciais, de seleção (ou de decisão) e de repetição (ou de iteração ou loop) são unidades básicas na escrita de algoritmos. Todas essas estruturas possuem condições a serem testadas; algumas realizam atribuição de variáveis, mas somente uma pode inicializar variáveis.

#### Comentário da questão

Vimos que um algoritmo pode ser constituído por três tipos de **estruturas de controle de fluxo**:

- **Estrutura sequencial**: é um bloco de comandos onde cada um deles é executado **passo a passo**, um após o outro;
- **Estrutura condicional** (ou **de seleção** ou **de decisão**): é um bloco de comandos que é executado ou não **dependendo de uma determinada condição** ser verdadeira ou falsa; e

- **Estrutura de repetição** (ou **de iteração** ou **de loop**): é um bloco de comandos executado **repetidas vezes até que uma condição seja alcançada**, encerrando-o e então o fluxo de execução dará continuidade ao restante das ações.

O seguinte trecho da questão está **certo**: “As estruturas de controle sequenciais, de seleção (ou de decisão) e de repetição (ou de iteração ou loop) são unidades básicas na escrita de algoritmos”.

Vamos analisa o restante:

- “Todas essas estruturas possuem condições a serem testadas”: **errado** porque, como vimos na citação acima, a estrutura sequencial não precisa testar condições;
- “algumas realizam atribuição de variáveis”: **errado**, pois todas as estruturas podem realizar atribuição de variáveis, ou seja, podem informar um valor para variáveis; e
- “somente uma pode inicializar variáveis”: **certo**, pelo fato que apenas a estrutura de repetição pode inicializar variáveis para serem utilizadas dentro da iteração da estrutura.

A questão ficaria correta dessa forma:

“As estruturas de controle sequenciais, de seleção (ou de decisão) e de repetição (ou de iteração ou loop) são unidades básicas na escrita de algoritmos. Uma dessas estruturas não possui condições a serem testadas; todas realizam atribuição de variáveis e apenas uma pode inicializar variáveis”.

Não se preocupem que vamos destrinchar esse assunto com mais calma. Por enquanto, fiquem com essas informações.

Gabarito: **ERRADO**.

## FEPESE 2010 SEFAZ/SC – Auditor Fiscal da Receita Estadual – Parte III – Tecnologia da Informação

Assinale a alternativa correta a respeito das variáveis e constantes, utilizadas em diversas linguagens de programação.

- A) O número de constantes deve ser menor ou igual ao número de variáveis em um programa.
- B) O número de constantes deve ser menor ou igual ao número de procedimentos em um programa.
- C) O número de constantes deve ser igual ao número de variáveis em um programa.
- D) O número de constantes independe da quantidade de variáveis em um programa.
- E) O número de constantes deve ser igual ao número de procedimentos em um programa.

### Comentário da questão

Quando criamos um algoritmo, o número de variáveis independe do número de constantes. Não há relação de forma alguma entre variáveis e constantes.

Gabarito: letra D.

## FCC 2008 Metrô/SP – Analista Trainee – Ciências da Computação

39 Em relação à lógica de programação, considere os pseudocódigos:

Algoritmo Alg1

```
salBase, salReceber, grat, imp: real
Inicio
Leia(salBase)
Grat ← salBase * 5/100
SalReceber ← salBase +grat – Imp
Imp ← SalReceber * 7/100
SalReceber ← SalReceber – imp
Escreva (salReceber)
Fim
```

Algoritmo 2

```
salBase, salReceber, Imp: real
Inicio
Leia(salBase)
SalReceber ← salBase+(salBase * 5/100 )
Imp ← SalReceber * 7/100
SalReceber ← SalReceber – imp
Escreva (salReceber)
Fim
```

É correto afirmar:

- A) Somente Alg1 tem consistência em sua representação e chega a um resultado.
- B) Ambos os algoritmos abordam o mesmo problema e chegam ao mesmo resultado.
- C) Somente Alg2 tem consistência em sua representação e chega a um resultado.
- D) O resultado da solução apresentada por Alg2 é maior do que a de Alg1.
- E) O resultado da solução apresentada por Alg2 é menor do que a de Alg1.

Comentário da questão

Para essa questão, vamos utilizar a técnica de teste de mesa ou teste chinês para sabermos o resultado de cada algoritmo. Para ambos algoritmos, o salário-base será de R\$ 1.000,00. Vamos nessa?

Teste de mesa para o algoritmo 1:

Linha	salBase	grat	imp	salReceber
Leia(salBase)	1000	-	-	-
Grat ← salBase * 5/100	1000	50	-	-
SalReceber ← salBase + grat - Imp	1000	50	-	1050
Imp ← SalReceber * 7/100	1000	50	73,5	1050
SalReceber ← SalReceber - imp	1000	50	73,5	976,5

Percebam que na linha  $SalReceber \leftarrow salBase + grat - Imp$ , a variável  $Imp$  não foi inicializada, então encaramos o seu valor como 0 (zero).

Vamos ao teste de mesa do algoritmo 2:

Linha	salBase	Imp	salReceber
Leia(salBase)	1000	-	-
$SalReceber \leftarrow salBase + (salBase * 5/100)$	1000	-	1050
$Imp \leftarrow SalReceber * 7/100$	1000	73,5	1050
$SalReceber \leftarrow SalReceber - imp$	1000	73,5	976,5

Vejam que o algoritmo 2 não possui a variável  $Grat$  como no algoritmo 1, porém, o cálculo do salário a receber acabou ficando a mesma coisa, ou seja, os dois algoritmos chegam ao mesmo resultado.

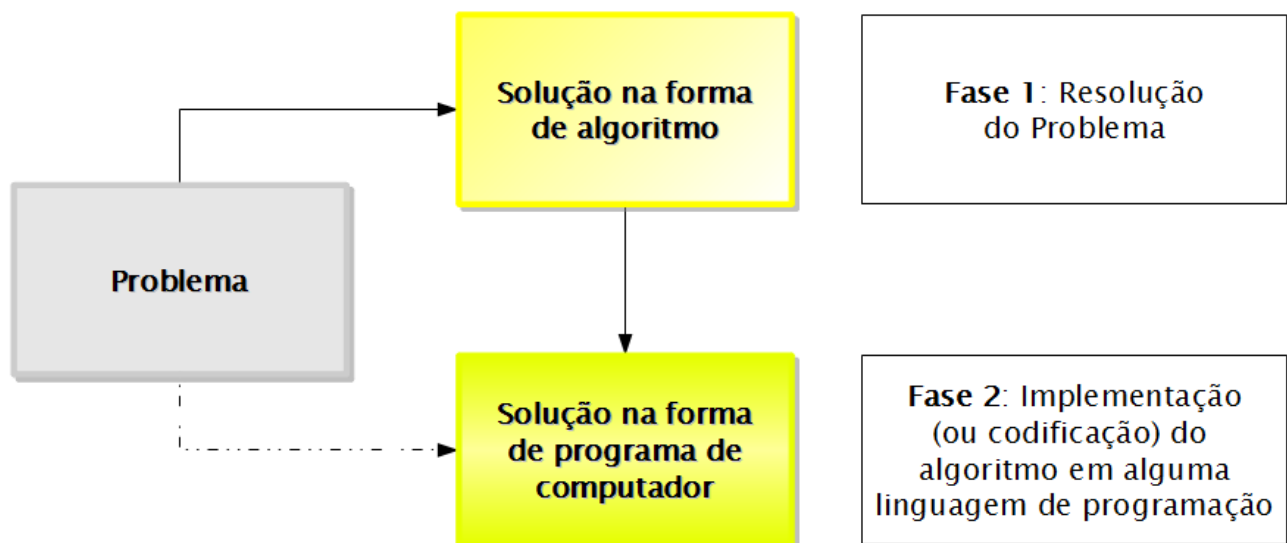
Gabarito: letra B.

### ESAF 2006 SUSEP – Analista Técnico – Tecnologia da Informação – Adaptada

47 Acerca dos conceitos fundamentais de lógica de programação e algoritmos, é incorreto afirmar que

- B) compreender o problema, selecionar um método de solução, descrever a solução passo a passo, validar o algoritmo, programá-lo e testá-lo, nesta seqüência, é uma proposta viável para analisar um problema.

Comentário da questão



A letra B citou uma série de passos para **analisar um problema**, exceto por "...programá-lo e testá-lo". Essa parte da programação seria a segunda fase de acordo com a imagem acima:

Primeiro, concentramos-nos em analisar e solucionar o problema na forma de algoritmo. Estaria o texto correto se não houvesse o seguinte trecho: "...programá-lo e testá-lo". Com isso, a letra B está **errada**.

Apenas para fechar, vamos recapitular nosso método simples para construção de um algoritmo:

- Compreendermos o problema a ser resolvido;
- Identificarmos e definirmos dos dados de entrada;
- Descrevermos detalhadamente os passos para processar ou transformar os dados de entrada para chegar ao objetivo do problema;
- Identificarmos e definirmos os dados de saída (objetivo do problema)
- Construímos o algoritmo que representa a descrição dos passos;
- Testarmos o algoritmo para possíveis correções que possam vir a ser necessárias na lógica proposta.

Gabarito: **ERRADO**.

## CESPE 2011 STM – Analista de Sistemas

Com relação a algoritmos e lógica de programação, julgue os itens a seguir.

**80** Nas estruturas de controle, tais como as estruturas de seleção simples, compostas ou encadeadas, é necessário verificar as condições para a realização de uma instrução ou sequência de instruções.

### Comentário da questão

Mesmo que não tenhamos visto ainda os conceitos de estruturas de seleção compostas ou encadeadas, saibam que é preciso verificar condições para que blocos de comandos sejam executados.

Gabarito: **CERTO**.

## FGV 2009 MEC – Arquiteto de Sistemas

**41** Analise o trecho de algoritmo a seguir, em pseudocódigo:

```
atribuir 'BRASIL-COPA2014' a STR;
atribuir 35 a GAMA;
atribuir 0 a BETA;
repetir
    se (resto da divisão de GAMA por 2 igual a 1)
        então imprimir (STR);
        atribuir GAMA-7 a GAMA;
    se GAMA=7 então atribuir 1 a BETA;
até que BETA = 1;
```

Após a execução, a variável STR será impressa uma quantidade de vezes igual a:

- A) 0
- B) 1
- C) 2
- D) 3
- E) 4

**Comentário da questão**

Vamos fazer nosso teste de mesa? 😊

Linha	STR	GAMA	BETA	Quantidade de impressão de STR
atribuir 'BRASIL - COPA 2014' a STR	'BRASIL - COPA 2014'	-	-	0
atribuir 35 a GAMA	'BRASIL - COPA 2014'	35	-	0
atribuir 0 a BETA	'BRASIL - COPA 2014'	35	0	0
repetir	'BRASIL - COPA 2014'	35	0	0
se (resto da divisão de GAMA por 2 igual a 1)	'BRASIL - COPA 2014'	35	0	0
imprimir STR	'BRASIL - COPA 2014'	35	0	1
atribuir GAMA - 7 a GAMA	'BRASIL - COPA 2014'	28	0	1
se GAMA = 7 então atribuir 1 a BETA	'BRASIL - COPA 2014'	28	0	1
até que BETA = 1	'BRASIL - COPA 2014'	28	0	1
repetir	'BRASIL - COPA 2014'	28	0	1
se (resto da divisão de GAMA por 2 igual a 1)	'BRASIL - COPA 2014'	28	0	1
atribuir GAMA - 7 a GAMA	'BRASIL - COPA 2014'	21	0	1
se GAMA = 7 então atribuir 1 a BETA	'BRASIL - COPA 2014'	21	0	1
até que BETA = 1	'BRASIL - COPA 2014'	21	0	1
repetir	'BRASIL - COPA 2014'	21	0	1
se (resto da divisão de GAMA por 2 igual a 1)	'BRASIL - COPA 2014'	21	0	1

imprimir STR	'BRASIL – COPA 2014'	21	0	2
atribuir GAMA – 7 a GAMA	'BRASIL – COPA 2014'	14	0	2
se GAMA = 7 então atribuir 1 a BETA	'BRASIL – COPA 2014'	14	0	2
até que BETA = 1	'BRASIL – COPA 2014'	14	0	2
repetir	'BRASIL – COPA 2014'	14	0	2
se (resto da divisão de GAMA por 2 igual a 1)	'BRASIL – COPA 2014'	14	0	2
atribuir GAMA – 7 a GAMA	'BRASIL – COPA 2014'	7	0	2
se GAMA = 7 então atribuir 1 a BETA	'BRASIL – COPA 2014'	7	1	2
até que BETA = 1	'BRASIL – COPA 2014'	7	1	2

Galera, a questão trouxe uma estrutura de repetição e outra de seleção e, por isso, algumas linhas foram repetidas e outras omitidas por condições não serem alcançadas. Para facilitar nossa visualização do teste de mesa, eu destaquei as repetições ímpares com fundo cinza.

Podemos concluir, com nosso teste de mesa, que toda vez que o valor de GAMA for igual a um número ímpar, dentre 35 a 14, decrescendo em 7, a variável STR será impressa. Quando GAMA chegar ao valor 7, BETA será incrementado em 1. Como a condição de saída da estrutura de repetição é que BETA seja 1, então o fluxo sairá da estrutura. Dessa forma, a variável STR é impressa duas vezes.

Gabarito: letra C.

## CESGRANRIO 2009 Cada da Moeda –Analista de Nível Superior – Negócios em TI

18 Analise o pseudocódigo a seguir.

- 1.var n: inteiro
- 2.escreva ("Digite um número inteiro:")
- 3.leia(n)
- 4.n<-n+5
- 5.escreva(n)

Considerando-se que o programa recebeu, como entrada, o valor 10, qual o resultado na tela da execução?

- A) 0
- B) 5
- C) 10
- D) 15



E) 20

### Comentário da questão

Linha	n
1	-
2	-
3	10
4	15
5	15

Uma observação que faço é que na linha 4, a variável n recebeu o valor anterior dela (10) acrescido de 5.

Gabarito: letra D.

### CESPE 2009 CEHAP/PB – Cargo 16: Programador

25 Considere o trecho de código a seguir.

```
INICIO
INTEIRO J, X;
J=1;
X=2;
ENQUANTO J<10 FAÇA
  X = X + 1;
  J = J + 2;
FIM ENQUANTO;
IMPRIMA X;
IMPRIMA J;
```

Ao final da execução do trecho de código acima, os valores de

- A) 7 e 11
- B) 6 e 12
- C) 8 e 11
- D) 9 e 12

### Comentário da questão

Linha	J	X
J=1	1	-
X=2	1	2
ENQUANTO J<10 FAÇA	1	2
X = X + 1	1	3
J = J + 2	3	3
ENQUANTO J<10 FAÇA	3	3
X = X + 1	3	4
J = J + 2	5	4
ENQUANTO J<10 FAÇA	5	4
X = X + 1	5	5
J = J + 2	7	5
ENQUANTO J<10 FAÇA	7	5
X = X + 1	7	6
J = J + 2	9	6
ENQUANTO J<10 FAÇA	9	6
X = X + 1	9	7
J = J + 2	11	7
FIM ENQUANTO	11	7
IMPRIMA X	11	7
IMPRIMA J	11	7

Enquanto o valor de J for menor que 10, as instruções contidas dentro da estrutura irão se repetirem, fazendo que as variáveis X e J acumulem valores.

Gabarito: letra A.

38 Considere o trecho de código a seguir.

```

INICIO
INTEIRO A, B, C, D;
A=10;
B=5;
C=0;
D=0;
ENQUANTO A ≥ 10 FAÇA
  C = A + B;
  D = B - A;
  A = A - 1;
FIM ENQUANTO;

IMPRIMA A;
IMPRIMA B;
IMPRIMA C;
IMPRIMA D;
    
```

Ao final da execução do trecho de código acima, os valores de A, B, C e D, são iguais, respectivamente, a

- A) 9, 5, 15 e 15.
- B) 10, 5, 15 e 15.
- C) 10, 5, 0 e 0.
- D) 10, 5, 15 e 5.

**Comentário da questão**

Linha	A	B	C	D
A=10	10	-	-	-
B=5	10	5	-	-
C=0	10	5	0	-
D=0	10	5	0	0
ENQUANTO A ≥ 10 FAÇA	10	5	0	0
C = A + B	10	5	15	0
D = B - A	10	5	15	-5
A = A - 1	9	5	15	-5
FIM ENQUANTO	9	5	15	-5

IMPRIMA A	9	5	15	-5
IMPRIMA B	9	5	15	-5
IMPRIMA C	9	5	15	-5
IMPRIMA D	9	5	15	-5

Enquanto o valor de A for maior que ou igual a 10, as instruções contidas dentro da estrutura irão se repetirem, fazendo que as variáveis C e D acumulem valores e a variável A tenha o seu valor subtraído de 1.

Gabarito: letra A.

<http://rogerioaraujo.wordpress.com>

