

# INTRODUÇÃO À PROGRAMAÇÃO

## 500 Algoritmos Resolvidos

- ↳ **Algoritmos resolvidos no cotidiano**
- ↳ **Algoritmos resolvidos com seleção e repetição**
- ↳ **Algoritmos resolvidos com vetores e matrizes**
- ↳ **Algoritmos resolvidos com funções**

N.Cham. 005.1 L85a 2002

Autor: Lopes, Anita

Título: Introdução à programação : 500



956299

Ac. 59639

CAMPUS

CD-ROM

Anita Lopes  
Guto Garcia

# Sumário

<b>Capítulo I</b>	<b>Conceitos iniciais</b>	1
<b>Capítulo 2</b>	<b>Variável, expressões, funções, atribuição, entrada e saída</b>	6
	Variável	6
	Expressões	9
	Funções	15
	Atribuição	23
	Comando de Saída	26
	Comando de Entrada	29
	Exercícios – Lista 1 ■ Leia, imprima, atribuição e funções	38
<b>Capítulo 3</b>	<b>Estruturas de seleção</b>	60
	Conceitos	60
	Sintaxes	61
	Ses Aninhados (Encaixados)	68
	Um pouco mais sobre variável string	73
	Algumas questões	75
	Alternativa de múltiplas escolhas	77
	Exercícios – Lista 2 ■ Estrutura do Se	78

<b>Capítulo 4</b>	<b>Estruturas de repetição: para, enquanto e faca-enquanto . . . . .</b>	124
	Estrutura do para . . . . .	124
	Exercícios – Lista 3 ■ Estrutura do PARA . . . . .	136
	Estrutura do enquanto . . . . .	178
	Estrutura do faca enquanto . . . . .	181
	Exercícios – Lista 4 ■ Enquanto e faca enquanto . . . . .	185
<b>Capítulo 5</b>	<b>Estruturas homogêneas: vetores e matrizes . . . . .</b>	266
	Conceito gerais . . . . .	266
	Ordenando vetores . . . . .	271
	Exercícios – Lista 5 ■ Lista de VETORES . . . . .	278
	Conceitos de matrizes . . . . .	332
	Triangulação de matrizes . . . . .	337
	Exercícios – Lista 6 ■ Lista de MATRIZES . . . . .	340
<b>Capítulo 6</b>	<b>Funções . . . . .</b>	394
	Conceito . . . . .	394
	Vantagens . . . . .	394
	Chamada da função . . . . .	395
	Estrutura de uma função . . . . .	396
	Localização das funções . . . . .	397
	Exercícios – Lista 7 ■ Lista de FUNÇÕES . . . . .	398
<b>Apêndice I</b>	<b>Interpretador UAL . . . . .</b>	452
<b>Apêndice II</b>	<b>Código ASCII . . . . .</b>	458
<b>Apêndice III</b>	<b>Raciocínio Lógico . . . . .</b>	460
	Bibliografia . . . . .	468

# Apresentação

Quando pensamos em produzir um livro sobre algoritmos, tínhamos a clareza de que ele deveria ter um diferencial; afinal, já existiam muitos no mercado.

Partimos do princípio de que ele deveria ter uma teoria e, principalmente, 500 algoritmos resolvidos, pois esta era a maior solicitação de nossos alunos: as soluções dos exercícios que passávamos.

Além disso, nossa maior preocupação era com a linguagem, que deveria ser a de uma pessoa leiga no assunto. Reunimos nossos materiais, anotamos dúvidas que surgiam durante as aulas e começamos nosso trabalho, que, aliás, teve muita participação de nossos alunos.

Ao longo desses quase três anos, fizemos muitas pesquisas e inventarmos muitos enunciados para chegar ao número estipulado por nós.

Decidimos que não iríamos além do assunto funções para que pudéssemos explorar ao máximo os assuntos abordados no livro.

Testamos todos os algoritmos até vetores no interpretador sugerido por nós, UAL e, todos que foram possíveis, no ILA.

Hoje, ao vermos nosso trabalho impresso, temos certeza de que era isso que gostaríamos de fazer e esperamos poder ajudar a você, leigo ou não.

Sabemos que um livro jamais substituirá a figura do professor, mas esperamos que o nosso faça você se sentir bem perto de nós, porque esse foi o nosso maior investimento.

ANITA LOPES & GUTO GARCIA

# Introdução

O aprendizado de Algoritmos nos cursos de graduação de Informática, Engenharia e Matemática, de acordo com da nossa experiência, é um processo extremamente difícil, tendo em vista o grande número de informações que os alunos precisam absorver em pouco tempo. Desta forma, temos observado uma desistência significativa da disciplina logo após a primeira avaliação.

As opiniões quanto ao ensino de algoritmos divergem muito, pois alguns professores defendem a tese de abstração, isto é, o aluno desenvolve no papel e não tem nenhum contato com a máquina.

Acreditamos que o aprendizado só se torna possível após a passagem do concreto para o abstrato; em outras palavras, o aluno precisa visualizar o que está fazendo e para isso o uso de um interpretador em português irá ajudá-lo nessa etapa inicial.

Nosso livro é resultado de um trabalho que vem sendo elaborado há algum tempo com nossos alunos. Nesses últimos anos, começamos a fazer todas as soluções dos exercícios passados para eles e percebemos que o aprendizado melhorou significativamente, uma vez que se torna inviável a resolução de um grande número de exercícios durante as aulas. Além disso, os alunos ficam muito inseguros nesse primeiro contato com a programação, pois é um processo muito solitário.

Este livro é voltado para o ensino inicial de algoritmos e procuramos fazer isso de uma maneira bem simples: o livro possui conceitos teóricos sobre algoritmos de uma forma bem rápida e resumida e 500 exercícios resolvidos, uma vez que acreditamos que o iniciante em programação precisa praticar muito para poder depois abstrair.

A sintaxe que usamos está mais parecida com a linguagem C e sugerimos o uso de um interpretador que roda sob ambiente Linux e cujas informações encontram-se no Apêndice I deste livro.

Muitas perguntas feitas por nossos alunos foram incorporadas a este livro porque podem ser também as suas dúvidas. Além disso, para explicar melhor o acompanhamento da execução do algoritmo, apresentaremos a saída no vídeo e a alocação da Memória Principal em relação às variáveis usadas para que você possa ir conhecendo um pouco mais sobre esse processo.

Os 500 algoritmos resolvidos estão divididos por assunto e organizados em 6 grandes blocos: o primeiro abrange algoritmos do cotidiano; no segundo, utilizamos somente algoritmos que usam funções, comandos de atribuição, de entrada e saída; no terceiro bloco, veremos o comando de seleção; no quarto bloco, os comandos de repetição; no quinto, utilizamos os algoritmos que manipulam vetores e matrizes, e no sexto, agrupamos algoritmos utilizando funções. Nos apêndices, falamos sobre uma ferramenta para testar os algoritmos no computador, código ASCII e apresentamos problemas de raciocínio lógico.

Não pule etapas. Procure fazer todos os exercícios de um capítulo antes de passar para outro. Estaremos com você em cada página, em cada comentário e temos certeza de que você aproveitará muito.

O aprendizado de algoritmos é algo apaixonante desde que você acredite e invista.

Qualquer dúvida disponibilizamos nossos e-mails:

Anita Lopes: [anitalml@iis.com.br](mailto:anitalml@iis.com.br)

Guto Garcia: [guto@cos.ufrj.br](mailto:guto@cos.ufrj.br)

## OS AUTORES



## Capítulo I

# Conceitos iniciais

- **Lógica de programação** é a técnica de encadear pensamentos para atingir determinado objetivo. O aprendizado desta técnica é necessário, para quem deseja trabalhar com desenvolvimento de sistemas e programas.
- **Algoritmo** é uma seqüência de passos finitos com o objetivo de solucionar um problema.

Quando nós temos um problema, nosso objetivo é solucioná-lo.

Algoritmo não é a solução de um problema, pois, se assim fosse, cada problema teria um único algoritmo. Algoritmo é um conjunto de passos (ações) que levam à solução de um determinado problema, ou então, é um caminho para a solução de um problema e, em geral, os caminhos que levam a uma solução são muitos.



O aprendizado de algoritmos não é uma tarefa muito fácil, só se consegue através de muitos exercícios. Este é o objetivo principal deste livro: possibilitar que você, a partir das soluções apresentadas, venha construir sua própria lógica de programação.

Todos nós, no dia-a-dia, nos deparamos com vários problemas. Quantas vezes já vimos um algoritmo e não sabíamos que aquela seqüência de passos chamava-se algoritmo. Um exemplo bem freqüente é quando queremos falar em algum telefone público. Aquilo que está escrito no telefone é um algoritmo. Veja a seguir um exemplo de um algoritmo do cotidiano.

- 1 – Retirar o telefone do gancho
- 2 – Esperar o sinal
- 3 – Colocar o cartão
- 4 – Discar o número
- 5 – Falar no telefone
- 6 – Colocar o telefone no gancho

O algoritmo é exatamente esse conjunto de passos que resolveu o problema de uma pessoa falar no telefone. É como se fôssemos ensinar uma máquina a fazer alguma tarefa específica.

Outro exemplo clássico é um algoritmo para resolver o problema de fritar um ovo que poderia estar escrito em forma de uma receita. A receita é um algoritmo, pois é formada de ações que devem ser tomadas para fritar o ovo. Como fritar um ovo é muito fácil, esta deve ser a primeira receita de fritar um ovo que vocês já leram.

### **algoritmo 1**

---

- 1 – pegar frigideira, ovo, óleo e sal
- 2 – colocar óleo na frigideira
- 3 – acender o fogo
- 4 – colocar a frigideira no fogo
- 5 – esperar o óleo esquentar
- 6 – colocar o ovo
- 7 – retirar quando pronto

Cada linha do algoritmo podemos chamar de uma instrução, logo, podemos dizer que um algoritmo é um conjunto de instruções.

■ **Instrução** indica a um computador uma ação elementar a ser executada.

Até as coisas mais simples podem ser descritas por um algoritmo. Por exemplo: “Mascar um chiclete”.

### **algoritmo 2**

---

- 1 – pegar o chiclete
- 2 – retirar o papel
- 3 – mastigar
- 4 – jogar o papel no lixo

Outros exemplos:

Comunicação

### algoritmo 3

#### Algoritmo para trocar lâmpadas

- 1 - se (lâmpada estiver fora de alcance)  
    pegar a escada;
- 2 - pegar a lâmpada;
- 3 - se (lâmpada estiver quente)  
    regar pano;
- 4 - Tirar lâmpada queimada;
- 5 - Colocar lâmpada boa;

### algoritmo 4

#### Algoritmo para o fim de semana

- 1 - vejo a previsão do tempo;
- 2 - se (fizer sol)  
    vou à praia;  
senão  
    vou estudar;
- 3 - almoçar
- 4 - ver televisão
- 5 - dormir

### algoritmo 5

#### Algoritmo para fazer um bolo simples

- 1 - pegar os ingredientes;
- 2 - se (roupa branca)  
    colocar avental;
- 3 - se (tiver batedeira)  
    bater os ingredientes na batedeira;  
senão  
    bater os ingredientes à mão;
- 4 - colocar a massa na forma;
- 5 - colocar a forma no forno;
- 6 - aguardar o tempo necessário;
- 7 - retirar o bolo;

### algoritmo 6

#### Algoritmo para descascar batatas

- 1 - pegar faca, bacia e batatas;
- 2 - colocar água na bacia;
- 3 - enquanto (houver batatas)  
    descascar batatas;

## **algoritmo 7**

---

*Algoritmo para fazer uma prova*

- 1 – ler a prova;
- 2 – pegar a caneta;
- 3 – enquanto ((houver questão em branco) e (tempo não terminou))faça  
    se (souber a questão)  
        resolvê-la;  
    senão  
        pular para outra;
- 4 – entregar a prova;

## **algoritmo 8**

---

*Algoritmo para jogar o jogo da forca:*

- 1 – escolher a palavra;
- 2 – montar o diagrama do jogo;
- 3 – enquanto ((houver lacunas vazias) e (corpo incompleto))faça  
    se (acertar uma letra)  
        escrever na lacuna correspondente;  
    senão  
        desenhar uma parte do corpo na forca;

## **algoritmo 9**

---

*Algoritmo para jogar o jogo da velha:*

- enquanto ((existir um quadrado livre ) e (ninguém perdeu(ganhou) o jogo))  
    espere a jogada do adversário, continue depois  
    se (existir um quadrado livre)  
        se (centro livre)  
            jogue no centro  
        senão  
            se (adversário tem 2 quadrados em linha com o terceiro  
                desocupado)  
                jogue neste quadrado desocupado  
            senão  
                se (há algum canto livre)  
                    jogue neste canto

## **algoritmo 10**

---

*Algoritmo para levar um leão, uma cabra e um pedaço de grama de um lado para outro de um rio, atravessando com um bote. Sabe-se que nunca o leão pode ficar sozinho com a cabra e nem a cabra sozinha com a grama.*

- 1 – Levar a grama e o leão
- 2 – Voltar com o leão

- 3 – Deixar o leão
- 4 – Levar a cabra
- 5 – Deixar a cabra
- 6 – Voltar com a grama
- 7 – Levar o leão e a grama

## DESAFIO

1) Fazer um algoritmo para levar 3 missionários e 3 canibais de um lado para outro de um rio, atravessando com um bote. Sabe-se que nunca pode ter mais missionários do que canibais porque senão os missionários catequizam os canibais. O que fazer para levar os 6 de uma margem para outra?

---

⇨ Ao formularmos um algoritmo, temos de ter clareza a respeito do aspecto estático dessa seqüência de ações escritas num pedaço de papel. O aspecto dinâmico só se evidencia ao executarmos essa seqüência no computador e daí, poderemos ter certeza se conseguimos ou não resolver o problema.

---

**Programa de Computador** nada mais é do que um algoritmo escrito numa linguagem de computador (C, Pascal, Fortran, Delphi, Cobol e outras). É a tradução para o inglês do algoritmo feito em português. O mais importante de um programa é a sua lógica, o raciocínio utilizado para resolver o problema, que é exatamente o algoritmo.



Introdução à programação em Python - I

## Capítulo 2

# Variável, expressões, funções, atribuição, entrada e saída

## VARIÁVEL

### Conceitos iniciais

Uma *variável* é um local na *memória principal*, isto é, um endereço que armazena um conteúdo. Em linguagens de alto nível, nos é permitido dar nome a esse endereço para facilitar a programação.

O conteúdo de uma variável pode ser de vários tipos: inteiro, real, caractere, lógico, entre outros. Normalmente, quando se ensina algoritmo, trabalha-se com os quatro tipos citados.

Uma vez definidos o nome e o tipo de uma variável, não podemos alterá-los no decorrer de um algoritmo. Por outro lado, o conteúdo da variável é um objeto de constante modificação no decorrer do programa, de acordo com o fluxo de execução do mesmo.

Em algoritmos, as variáveis serão definidas no início, por meio do comando definido:

tipo da variável

nome da variável

;

```
int a;  
real b;  
string nome;  
logico r;
```

## OBSERVAÇÃO

Quando formos dar nome às variáveis, se faz necessário seguirmos algumas regras. É bom ressaltar que estas regras irão variar de acordo com a linguagem. As seguintes regras:

1. O primeiro caractere é uma letra.
2. Se houver mais de um caractere, só poderemos usar: letra ou algarismo.
3. Nomes de variáveis escritas com letras maiúsculas serão diferentes de letras minúsculas. Lembre-se: media é diferente de MEDIA.
4. Nenhuma palavra reservada poderá ser nome de uma variável.

Nomes Válidos	Nomes Não-Válidos
media, alt, a2, PESO	2w -> começa por algarismo media_aluno -> o caractere sublinha não é permitido peso do aluno -> o caractere espaço não é permitido

Um dos objetivos de se declarar uma variável no início do algoritmo é para que seja alocada (reservada) uma área na memória (endereço de memória) para a variável. Outro objetivo da declaração de variáveis é que, após a declaração, o algoritmo sabe os tipos de operações que cada variável pode realizar; explicando: algumas operações só podem ser realizadas com variáveis do tipo inteiro, outras só podem ser realizadas com variáveis dos tipos inteiro ou real, outras só com variáveis de caractere etc.

## Tipos de variáveis

### Numérica

Variáveis numéricas são aquelas que armazenam dados numéricos, podendo ser divididas em duas classes:

*int*

Os números inteiros são aqueles que **não** possuem componentes decimais ou fracionários, podendo ser positivos ou negativos.

As variáveis compostas com esses números são chamadas de **VARIÁVEIS INTEIRAS**. Os elementos pertencentes aos conjuntos N e Z, apesar de serem

representáveis na classe dos números reais, são classificados como variáveis numéricas inteiras por não possuírem parte fracionária.

Como exemplo de números inteiros temos:

- 12      número inteiro positivo
- 12     número inteiro negativo

---

↳ Normalmente, uma variável do tipo inteira poderá ocupar **1, 2 ou 4 bytes** na MP.

---

### *real*

Os números reais são aqueles que podem possuir componentes decimais ou fracionários, podendo também ser positivos ou negativos.

As variáveis compostas com estes números pertencentes aos conjuntos dos números reais são chamadas de **VARIÁVEIS REAIS**.

Como exemplos de números reais temos:

- 24.01     número real positivo com duas casas decimais
- 144.       número real positivo com zero casa decimal
- 13.3      número real negativo com uma casa decimal
- 0.0        número real com uma casa decimal

---

↳ Normalmente, uma variável do tipo real poderá ocupar **4 ou 8 bytes** na MP.

---

### *string*

Também conhecida como caractere, alfanumérica ou literal. Esse tipo de variável armazena dados que contêm letras, dígitos e/ou símbolos especiais.

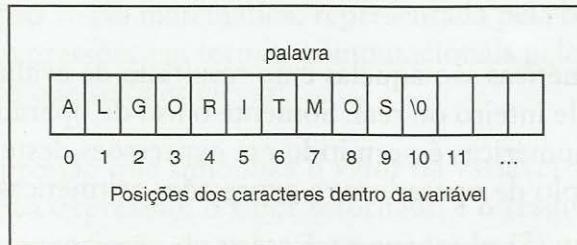
Como exemplos de constantes string temos:

- “Maria”    string de comprimento 5
- “123”      string de comprimento 3
- “0”         string de comprimento 1
- “A”         string de comprimento 1

O número de bytes possíveis para armazenamento de uma variável **string** dependerá da linguagem, mas o mais importante é entender que uma variável string é armazenada na MP como sendo uma **matriz linha**. Observe o trecho de algoritmo a seguir e suponha que na entrada de dados foi digitado: ALGORITMOS.

```
string palavra;  
leia palavra;
```

Memória Principal (MP)



Armazenamento  
de uma variável  
caractere no UAL

Em algumas linguagens, a numeração poderá começar com 1.

### OBSERVAÇÃO

*Não confundir: caractere que se encontra na posição 3 com 3º caractere:*

No exemplo acima, temos:

- caractere que se encontra na posição 3 : O
- 3º caractere : G

### lógico

Também conhecido como booleano. É representado no algoritmo pelo dois únicos valores lógicos possíveis: verdadeiro ou falso. Porém, é comum encontrar em outras referências outros tipos de pares de valores lógicos como: sim/nao, 1/0, true/false, verdadeiro/falso.

Como exemplos de constantes lógicas temos:

verdadeiro	Valor lógico verdadeiro
falso	Valor lógico falso

### OBSERVAÇÃO

*As variáveis quando são declaradas, dependendo da linguagem, não têm nenhum valor atribuído; portanto, no início, atribua valores a todas as variáveis.*

## EXPRESSÕES

O conceito de expressão em termos computacionais está intimamente ligado ao conceito de expressão (ou fórmula) matemática, onde um conjunto de variáveis e constantes numéricas relaciona-se por meio de operadores compondo uma fórmula que, uma vez avaliada, resulta num valor. As expressões dividem-se em:

## Aritméticas

Expressões aritméticas são aquelas cujo resultado da avaliação é do tipo numérico, seja ele inteiro ou real. Somente o uso de operadores aritméticos e variáveis numéricas é permitido em expressões deste tipo.

Como exemplo de operadores e expressões aritméticas temos:

**Soma** Na matemática, representada pelo sinal + e, em expressões em termos computacionais, pelo mesmo sinal.

$A + B$  Expressão que simboliza a soma do valor de duas variáveis.

$2 + 3$  Nessa expressão, o valor retornado é a soma dos valores dados, isto é, 5.

**Subtração** Na matemática, representada pelo sinal – e, em expressões em termos computacionais, pelo mesmo sinal.

$A - B$  Expressão que simboliza a subtração do valor de duas variáveis.

$3 - 2$  Nessa expressão, o valor retornado é o resto, isto é, 1.

**Multiplicação** Na matemática, representada pelos sinais  $\times$  ou . e, em expressões em termos computacionais, pelo sinal \*.

$B * D$  Expressão que simboliza a multiplicação do valor de duas variáveis.

$3 * 2$  Nessa expressão, o valor retornado é o produto dos valores dados, isto é, 6.

**Divisão** Na matemática, representada pelo sinal  $\div$  e, em expressões computacionais, pelo sinal /.

$A / B$  Expressão que simboliza a divisão do valor de duas variáveis.

$6 / 2$  Nessa expressão, o valor retornado é a divisão dos valores dados, que, no caso, será equivalente a 3.

$5 / 2$  Nessa expressão, o valor retornado é a divisão dos valores dados, que, no caso, será equivalente a 2.5.

### OBSERVAÇÃO

■ Normalmente, as linguagens de programação assumem que a divisão é uma operação que retorna um valor REAL. Atenção especial, portanto, para variáveis que receberão o resultado de uma divisão.

■ Em algumas linguagens, quando se divide dois números inteiros, o resultado será um inteiro.

**Exponenciação** Na matemática, representada pela base e por um expoente e em expressões em termos computacionais pelo sinal (\*\* ou ^) mais o número que se quer elevar.

- A \*\* 2 Expressão que simboliza o valor da variável ao quadrado.  
3 ^ 2 Nessa expressão, o valor retornado é o resultado da exponenciação do valor 3 ao quadrado (2) que, no caso, será equivalente a 9.  
2 \*\* 3 Nessa expressão, o valor retornado é o resultado da exponenciação do valor 2 ao cubo (3), que no caso será equivalente a 8.00.

### OBSERVAÇÃO

Normalmente, as linguagens oferecem um dos operadores citados, mas usaremos os dois e a diferença será explicada a seguir:

- \*\* - exponenciação com resultado Real
- ^ - exponenciação com resultado Inteiro, fazendo arredondamento.

- 8 \*\* 3 A resposta seria 512.00  
8 ^ 3 A resposta seria 512  
8.5 \*\* 3 A resposta seria 614.125  
8.5 ^ 3 A resposta seria 614

### ↳ Radiciação pela potência

$$\sqrt[\text{índice}]{\text{radicando}} = \text{radicando}^{1/\text{índice}}$$

$$\text{Exemplo: } \sqrt[3]{512} = 512^{1/3} \rightarrow 512^{**}(1/3)$$

**% – resto** Em outras linguagens, conhecido como mod. É usado em expressões em termos computacionais quando se deseja encontrar o resto da divisão de *dois números inteiros*.

- K % Y Expressão que simboliza a intenção de achar o resto da divisão do valor da variável K pelo valor da variável Y.  
5 % 2 Nessa expressão, o valor retornado é o resto da divisão do primeiro pelo segundo número, que, no caso, será equivalente a 1.

**7 % 4** Nessa expressão, o valor retornado é o resto da divisão do primeiro pelo segundo número, que, no caso, será equivalente a 3.

**div – divisão inteira** É usada em expressões em termos computacionais quando se deseja encontrar o quociente da divisão de dois números inteiros.

**A div B** Expressão que simboliza a intenção de achar o valor do divisor na divisão do valor da variável A pelo valor da variável B.

**5 div 2** Nessa expressão, o valor retornado é o coeficiente da divisão do primeiro pelo segundo número, que, no caso, será equivalente a 2.

↳ O operador **div** necessita que, antes e depois dele, pressionemos a barra de espaço.

## Relacional

Uma expressão relacional, ou simplesmente relação, é uma comparação realizada entre dois valores de mesmo *tipo básico*. Estes valores são representados na relação através de constantes, variáveis ou expressões aritméticas.

Como exemplos de operadores relacionais matematicamente conhecidos temos:

Operador	Matemática	Usaremos
Igual	=	==
Diferente	≠	< >
Maior	>	>
Menor que	<	<
Maior ou Igual a	≥	≥ =
Menor ou Igual a	≤	≤ =

Como exemplos de expressões relacionais temos:

$A < > B$  A diferente de B

$X == 1$  X igual a 1

$7 > 6$  7 maior que 6

$8 < 9$  8 menor que 9

$1 <= Y$  1 menor ou igual ao valor da variável Y

$4 >= W$  4 maior ou igual ao valor da variável W

## Lógica ou booleana

Denomina-se expressão lógica a expressão cujos operadores são lógicos e cujos operandos são relações, constantes e/ou variáveis do tipo lógico.

Como exemplo de operadores lógicos, matematicamente conhecidos temos:

Operador	Matemática	Usaremos
conjunção	e	&&
disjunção	ou	
negação	nao	!

### Tabela verdade do operador &&

Suponha duas perguntas feitas a quatro pessoas. Se a resposta do candidato for falsa, deverá falar 0, caso contrário, falará 1.

Suponha também que só será chamado para entrevista o candidato que dominar as duas linguagens.

Você conhece a linguagem C?	Você conhece a linguagem PASCAL?	Saída
0	0	0
0	1	0
1	0	0
1	1	1

Nesse exemplo, somente o quarto candidato seria chamado para a entrevista, pois o operador && (e) só considera a expressão como verdadeira se todas as expressões testadas forem verdadeiras.

### Tabela verdade do operador ||

Suponha duas perguntas feitas a quatro pessoas. Se a resposta do candidato for falsa, deverá falar 0, caso contrário, falará 1.

Suponha também que será chamado para entrevista o candidato que dominar pelo menos uma linguagem.

Você conhece a linguagem C++?	Você conhece a linguagem JAVA?	Saída
0	0	0
0	1	1
1	0	1
1	1	1

Nesse exemplo, somente o primeiro candidato *não* seria chamado para a entrevista, pois o operador `||` (ou) considera a expressão como verdadeira se *pelo menos* uma expressão testada for verdadeira.

### OBSERVAÇÃO

1. A seguir, relacionamos os critérios de precedência dos operadores. Lembre-se de que algumas linguagens não obedecem a estes critérios.
2. Se precisarmos alterar esta hierarquia, usaremos os parênteses.

Hierarquia	
<i>primeiro</i>	<i>parênteses e funções</i>
<i>segundo</i>	<i>potência e resto</i>
<i>terceiro</i>	<i>multiplicação e divisão</i>
<i>quarto</i>	<i>adição e subtração</i>
<i>quinto</i>	<i>operadores relacionais</i>
<i>sextº</i>	<i>operadores lógicos</i>

### Tabela verdade do operador !

Observe a tabela a seguir e as afirmativas:

- A cor da camisa A não é azul.
- A cor da camisa B não é amarela.

Camisa	Cor	SAÍDA
A	Azul	falso
B	Verde	verdadeiro

O operador `!` (não) inverte a saída.

Considere  $a$ ,  $b$  e  $c$  variáveis numéricas, e cor uma variável string. Como exemplos de expressões lógicas temos:

$a + b == 0 \ \&\& \ c < > 1$	Essa expressão verifica se o resultado da soma dos valores das variáveis $a$ e $b$ é igual a 0 e ( $\&\&$ ) se o valor da variável $c$ é diferente de 1. O resultado será considerado verdadeiro se as <i>duas</i> expressões relacionais foram verdadeiras.
$Cor == "azul" \    \ a * b > c$	Essa expressão verifica se o conteúdo armazenado na variável $cor$ é azul ou ( $  $ ) se o resultado do produto dos valores variáveis $a$ e $b$ é maior do que o valor armazenado na variável $c$ . O resultado será considerado verdadeiro se, <i>pelo menos uma</i> das expressões relacionais for verdadeira.

O resultado obtido de uma avaliação de uma expressão lógica é sempre um valor lógico, isto é, falso ou verdadeiro. Por esse motivo, pode-se considerar uma única relação como sendo uma expressão lógica.

## FUNÇÕES

O conceito de função em termos computacionais está intimamente ligado ao conceito de função (ou fórmula) matemática, onde um conjunto de variáveis e constantes numéricas relaciona-se por meio de operadores, compondo uma fórmula que, uma vez avaliada, resulta num valor. As expressões se dividem em:

### Numéricas

Funções numéricas são aquelas cujo resultado da avaliação é do tipo numérico, seja ele inteiro ou real. Somente podem ser efetuadas entre números propriamente apresentados ou variáveis numéricas.

Como exemplos de funções numéricas temos:

**pi** Função que resulta no valor 3.14159265, Sem argumentos.

**sen(x)** Função que resulta no valor do seno de um ângulo qualquer em radianos.

---

↪ Antes de aplicar a função **sen**(ang), deve-se transformar o Ângulo em Graus para Ângulo Radiano com a seguinte fórmula matemática:  $ang * 3.14159265 / 180$  (ângulo multiplicado por 3.14159265 e o resultado dividido por 180). E logo depois se pode aplicar a função.

---

A constante 3.14159265 será predefinida: pi.

Logo, teremos:

```
angrad <- ang * pi / 180;
imprima sen(angrad);
```

Dessa forma, podemos observar que somente usamos a função **sen(x)** depois de transformar o ângulo, dado em graus, em ângulo radiano.

---

↪ Normalmente, as linguagens de programação assumem que a função **sen()** é uma função que retorna um valor REAL. Atenção especial, portanto, para variáveis que receberão como conteúdo expressões que envolvam essa função.

---

**cos(x)** Função que resulta no valor do co-seno de um de um ângulo qualquer em radianos.

Logo, teremos:

```
angrad <- ang * pi / 180;
imprima cos(angrad);
```

---

↪ Antes de aplicar a função **cos(ang)**, deve-se transformar o Ângulo em Graus para Ângulo Radiano com a seguinte formula matemática:  $ang * 3.14159265/180$  (ângulo multiplicado por 3.14159265 e o resultado dividido por 180). E logo depois se pode aplicar a função.

---

Dessa forma, podemos observar que somente usamos a função **cos(x)** depois de transformar o ângulo, dado em graus, em ângulo radiano.

---

↪ Normalmente, as linguagens de programação assumem que a função **cos()** é uma função que retorna um valor REAL. Atenção especial, portanto, para variáveis que receberão como conteúdo expressões que envolvam essa função.

---

**tan(x)** Função que resulta no valor da tangente de um ângulo qualquer em radianos.

Logo, teremos:

```
angrad <- ang * pi / 180;
imprima tan(angrad);
```

---

↳ Antes de aplicar a função ***tan***(ang), deve-se transformar o Ângulo em Graus para Ângulo Radiano com a seguinte fórmula matemática:  $\text{ang} * 3.14159265/180$  (ângulo multiplicado por 3.14159265 e o resultado dividido por 180). E logo depois se pode aplicar a função.

---

Dessa forma, podemos observar que somente usamos a função ***tan(x)*** depois de transformar o ângulo, dado em graus, em ângulo radiano.

---

↳ Normalmente, as linguagens de programação assumem que a função ***tan()*** é uma função que retorna um valor REAL. Atenção especial, portanto, para variáveis que receberão como conteúdo expressões que envolvam essa função.

---

***abs(x)*** Função que resulta no valor absoluto de um número qualquer.

$\text{abs}(7)$  Neste caso, a resposta fornecida seria 7  
 $\text{abs}(-7)$  Neste caso, a resposta fornecida seria 7

***exp(x)*** Função que resulta no valor do número e (base do logaritmo neperiano) elevado a um número qualquer.

$\text{exp}(3)$  Nesse caso, seria o mesmo que  $e^3 \rightarrow 2.71828182846^{**} 3$   
 $\text{exp}(2)$  Nesse caso, seria o mesmo que  $e^2 \rightarrow 2.71828182846^{**} 2$

Dessa forma, podemos observar que a função ***exp(x)*** se refere à base e (base do logaritmo neperiano: 2.71828182846) elevada ao número citado entre os parênteses da função ***exp(x)***.

---

↳ Normalmente, as linguagens de programação assumem que a função ***exp()*** é uma função que retorna um valor REAL. Atenção especial, portanto, para variáveis que receberão como conteúdo expressões que envolvam essa função.

---

***log(x)*** Função que resulta no valor do logaritmo neperiano de um número qualquer.

$\text{log}(3)$  Nesse caso, seria: 1.09861

Dessa forma, podemos observar que o logaritmo ao qual a função ***log(x)*** se refere é sempre denominado pela base e.

↳ Na prática, poderá ser necessário calcular o logaritmo em outra base e, para isso, você deverá fazer a conversão entre bases logarítmicas, usando a seguinte propriedade:

$$\log_B^A = \frac{\log_x^A}{\log_x^B}$$

Se considerarmos que a base dos logaritmos naturais (**e**) será usada, teremos:

**log(A) / log(B)**

↳ Normalmente, as linguagens de programação assumem que a função **log()** é uma função que retorna um valor REAL. Atenção especial, portanto, para variáveis que receberão como conteúdo expressões que envolvam essa função.

**raiz(x)** Função que resulta no valor da raiz quadrada de um número positivo.

raiz(4) Nesse caso, seria o mesmo que  $\sqrt{4} = 2$

raiz(9) Nesse caso, seria o mesmo que  $\sqrt{9} = 3$

Dessa forma, podemos observar que a função **raiz(x)** sempre fornece a raiz quadrada do argumento que sempre será positivo.

↳ Normalmente, as linguagens de programação assumem que a função **raiz()** é uma função que retorna um valor REAL. Atenção especial, portanto, para variáveis que receberão como conteúdo expressões que envolvam essa função.

## Funções de Conversão de Tipos

1) **realint(número real)** Função que converte um número real em inteiro.

realint(11.5) Nesse caso, retornaria 12.

realint(12.51) Nesse caso, retornaria 13.

↳ Veja considerações sobre a função **realint** no Apêndice.

2) **intreal**(número inteiro) Função que converte um número inteiro em real.

- intreal(11) Nesse caso, retornaria 11.0.  
intreal(12) Nesse caso, retornaria 12.0.

## Caracter

Funções caracter são aquelas cujo resultado da avaliação é do tipo caracter. Somente podem ser efetuadas entre caracteres propriamente apresentados ou variáveis literais do tipo caracter.

Como exemplo de funções caracter temos:

**strtam(string)** Função que retorna número de caracteres de uma string.

- strtam("rio") O resultado seria 3.  
strtam(nome) Nesse caso o resultado será o tamanho do conteúdo da variável nome.

Dessa forma, podemos observar que a função strtam(x) sempre retorna o número de caracteres de uma string ou variável string.

---

↳ Se o tamanho de uma variável string for armazenado em uma variável, o tipo dessa variável deverá ser **int**.

---

**strelem(string, pos)** Função que retorna o elemento da string que se encontra na posição indicada na função como *pos*.

Suponha a variável palavra

palavra										
A	L	G	O	R	I	T	M	O	S	ó
0	1	2	3	4	5	6	7	8	9	10
Posições dos caracteres dentro da variável										

- strelem(palavra,2) O resultado seria a letra G.  
strelem(palavra,0) O resultado seria a letra A.  
strelem(palavra,5) O resultado seria a letra I.  
strelem(palavra,10) O resultado seria uma mensagem de erro  
indicando argumento inválido.

---

↳ A variável ou constante *pos*, presente na função, representa a posição do caractere dentro da variável, porém não se esquecendo que a primeira posição é 0(zero).

---

**strprim(string)** Função que retorna o primeiro elemento da string.

Suponha a variável palavra

palavra										
A	L	G	O	R	I	T	M	O	S	\0
0	1	2	3	4	5	6	7	8	9	10

strprim(palavra)

O resultado seria a letra A, pois a função reconhece que o primeiro caractere se encontra na posição 0 (zero).

**strnprim(string, n)** Função que retorna os n primeiros elementos da string, incluindo a posição 0 (zero).

Suponha a variável palavra

palavra										
A	L	G	O	R	I	T	M	O	S	\0
0	1	2	3	4	5	6	7	8	9	10

strnprim(palavra, 2)

O resultado seria AL, pois a função entende que os dois primeiros elementos estão nas posições 0 (zero) e 1 (um).

strnprim(palavra, 4)

O resultado seria ALGO, pois a função entende que os quatro primeiros elementos estão nas posições 0 (zero), 1 (um), 2 (dois) e 3 (três).

**strresto(string)** Função que retorna todos os elementos da string, exceto o primeiro.

Suponha a variável palavra

palavra										
A	L	G	O	R	I	T	M	O	S	\0
0	1	2	3	4	5	6	7	8	9	10

strresto(palavra)

O resultado seria LGORITMOS.

Nesse caso, a função reconhece que o primeiro elemento se encontra na posição 0 (zero).

**strlult(string)** Função que retorna o último elemento da string.

Suponha a variável palavra

palavra										
A	L	G	O	R	I	T	M	O	S	\0
0	1	2	3	4	5	6	7	8	9	10

strlult(palavra)

O resultado seria a letra S.

**strnresto(string, n)** Função que retorna os elementos da string após os n primeiros.

Suponha a variável palavra

palavra										
A	L	G	O	R	I	T	M	O	S	\0
0	1	2	3	4	5	6	7	8	9	10

strnresto (palavra, 2)

O resultado seria GORITMOS.

---

↪ Nesse caso, a função reconhece que os dois primeiros elementos se encontram nas posições 0 (zero) e 1 (um).

---

**strcopia(string)** Função que copia a string. Deverá ser usada com o comando de atribuição.

a <- "UNESA";      A string UNESA é armazenada na variável a.  
b <- strcopia(a);    O conteúdo da variável a é copiado para variável b.  
a <- "UNESA";      O resultado seria idêntico ao anterior.  
b <- a;

### OBSERVAÇÃO

Em algumas linguagens de programação, não é permitido usar o comando de atribuição a seguir:

b <- a;

Mas, em compensação, há uma outra alternativa que possibilita uma variável string receber o conteúdo de outra variável string:

**strcopia(string1,string2):**  
strcopia( b, a );

**strcmp(string1,string2)** Função que resulta na comparação por ordem alfabética de duas strings (string1 e string2) retornando:

“igual” se forem iguais.

“menor” se string1 vier antes de string2.

“maior” se string1 vier depois de string2.

strcmp(“maria”, “maria”) Nesse caso, o valor retornado seria “igual”.

strcmp(“aline”, “alex”) Nesse caso, o valor retornado seria “maior”.

strcmp(“carina”, “simone”) Nesse caso, o valor retornado seria “menor”.

strcmp(a, b) Nesse caso, seriam comparados os conteúdos das variáveis a e b.  
O resultado poderia ser: “maior”, “igual” ou “menor”.

---

↳ Na maioria das linguagens, os resultados das comparações serão: 0 ou um número negativo ou um número positivo.

---

Explicação:

Observe as figuras a seguir, cujas letras estão representadas pelos respectivos códigos ASCII:

a	l	i	n	e	a	l	e	x
97	108	105	110	101	97	108	101	119

quando se usa: ... strcmp(“aline”, “alex”) ... , na verdade, compara-se o 1º código de aline com o 1º código de alex; como são iguais, compara-se o 2º código de aline com o 2º código de alex; como são iguais, compara-se o 3º código de aline com o 3º código de alex mas, nessa comparação (105 – 101), o resultado foi maior do que zero (0), logo entende-se que aline, na ordem alfabética, vem depois de alex.

---

↳ Normalmente, essa função será usada com estruturas de teste que serão vistas mais adiante.

---

**strconcat(string1, string2)** Função que resulta na cópia do valor contido em uma string2 para o final da string1.

a <- "ANITA &"; A string ANITA & é armazenada na variável a.

b <- "GUTO"; A string GUTO é armazenada na variável b.

c <- strconcat(a, b); A variável c recebe o conteúdo: ANITA&GUTO.

↳ Os argumentos das funções strings deverão ser variáveis ou constantes. Funções não são permitidas.

## ATRIBUIÇÃO

É a principal forma de se armazenar um dado em uma variável. Esse comando permite que você forneça um valor a uma variável, onde o tipo desse valor tem de ser compatível com a variável.

identificador	<-	expressão	;
---------------	----	-----------	---

Legenda:

identificador é o nome da variável à qual está sendo atribuído um valor.

<- é o símbolo de atribuição, formado pelo sinais < e -.

expressão pode ser uma expressão aritmética, uma expressão lógica ou literal cuja avaliação (resultado) é atribuída ao identificador (variável).

finaliza o comando.

Exemplo 1 : x <- 10 ;

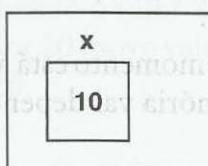
Como se lê ?

A variável x recebe o valor 10 ou x recebe o valor 10.

O que faz o computador ?

Nesse momento, na memória do computador, onde já estava alocado um espaço para a variável x (realizado na declaração de variáveis), essa variável recebe o valor 10.

### Memória Principal (MP)



Exemplo 2 :  $x \leftarrow a + b ;$

Como se lê?

A variável  $x$  recebe o resultado do conteúdo da variável  $a$  somado ao conteúdo da variável  $b$  ou  $x$  recebe o valor de  $a$  somado a  $b$  ou, ainda,  $x$  recebe  $a + b$ . Lembre-se de que as operações aritméticas são realizadas pela UAL.

O que faz o computador?

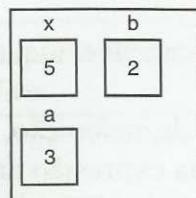
Nesse momento, na memória do computador, onde já estava sendo alocado espaço para as variáveis  $a$ ,  $b$  e  $x$ , o conteúdo da variável  $x$  vai receber a soma do conteúdo de  $a$  e  $b$ .

---

→ No comando de atribuição em que o valor é representado por uma expressão aritmética, lógica ou literal, estas devem ser avaliadas em primeiro lugar para que, então, o resultado obtido seja armazenado na variável.

---

Memória Principal (MP)

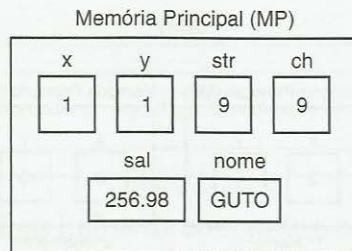


Exemplo 3:

$y \leftarrow 1;$	$y$ recebe o valor 1.
$x \leftarrow y;$	$x$ recebe o conteúdo que está em $y$ ; mas como $y$ vale 1, $x$ vai receber 1, que é o conteúdo de $y$ .
$sal \leftarrow 256.98;$	$sal$ recebe o valor 256.98.
$nome \leftarrow "GUTO";$	a variável $nome$ recebe a string "GUTO".
$chr \leftarrow "g";$	a variável $chr$ recebe o caractere "g"
$str \leftarrow chr;$	$str$ recebe o conteúdo de $chr$ que é "g".

Então, podemos resumir o exemplo acima como:  $x$  e  $y$  são duas variáveis inteiras;  $sal$  é uma variável do tipo real;  $nome$  é uma variável do tipo caractere;  $ch$  e  $str$  são variáveis do tipo char, e chave é uma variável do tipo lógica.

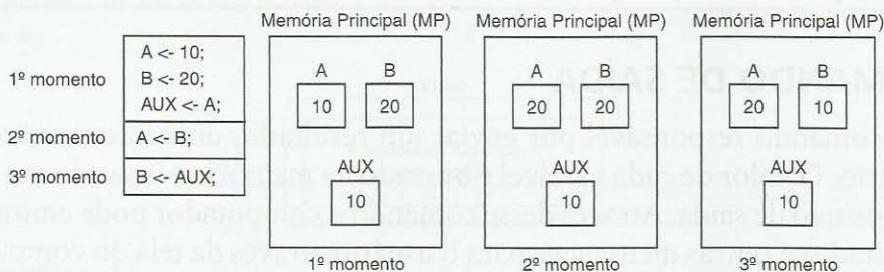
O mapa da memória nesse momento está assim. Podemos verificar que o tamanho do espaço na memória vai depender do tipo de variável.



Conclusão:

O comando de atribuição é muito importante em algoritmos. O próximo exemplo nos mostra isso.

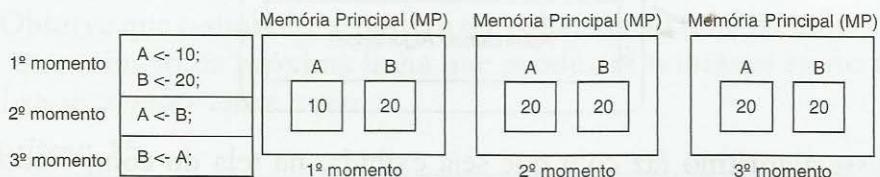
Exemplo 4:



Qual o objetivo do algoritmo acima?

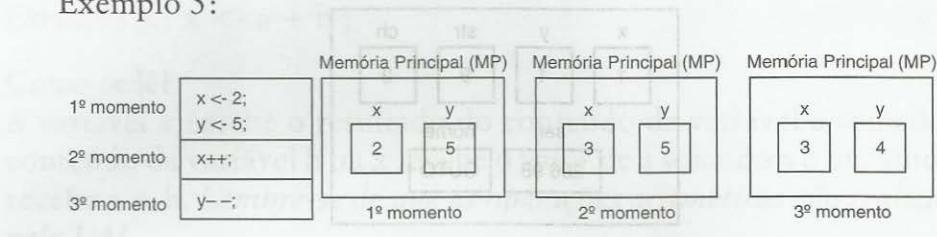
O conteúdo das variáveis A e B é trocado, isto é, no final do algoritmo, a variável A está com o valor 20 e a variável B está com o valor 10. Tivemos de utilizar uma variável auxiliar (AUX) que guarda o valor da variável A.

Para ficar mais clara a importância da variável auxiliar, observe a próxima figura e veja o que aconteceria se não tivéssemos feito uso dela:



Como pode ser visto, quando a variável A recebe o valor de B, a variável A perde o seu valor 10, recebendo o valor 20; depois, quando B receber o valor de A, B vai receber o valor 20 (novo valor de A). No final as variáveis A e B vão ter o mesmo valor 20.

### Exemplo 5:



Os operadores `++` e `--` são operadores de incremento e decremento. Eles são usados para realizar operações de adição e subtração e são similares à atribuição para variáveis inteiras.

`x++ ;` É equivalente a:  $x \leftarrow x + 1$ ;

`y-- ;` É equivalente a:  $y \leftarrow y - 1$ ;

## COMANDO DE SAÍDA

É o comando responsável por enviar um resultado, uma informação ao usuário. O valor de cada variável é buscado na memória e inserido em um dispositivo de saída. Através desse comando o computador pode emitir os resultados e outras mensagens para o usuário através da tela do computador ou uma impressora.

**imprima** expressão ou variável ou constantes ;

### algoritmo 11

```
prog impl
    imprima "Aprendendo Algoritmo!!!";
fimprog
```

Aprendendo Algoritmo!!!

Esse algoritmo faz com que seja exibida, na tela do computador, a mensagem: Aprendendo Algoritmo!!!

### algoritmo 12

```
prog imp2
    imprima "Aprendendo Algoritmo !!!";
    imprima "Com Anita e Guto";
fimprog
```

Vídeo

Aprendendo Algoritmo!!!Com Anita e Guto

Embora tenhamos duas linhas de comandos, o que nos levaria a pensar que teríamos duas linhas no vídeo, o interpretador só alimenta linha se assim especificarmos através do símbolo `\n`. Esse símbolo é uma string e deverá vir entre aspas.

### algoritmo 13

```
prog imp3
    imprima "Aprendendo Algoritmo!!!";
    imprima "\nCom Anita e Guto";
fimprog
```

Vídeo

Aprendendo Algoritmo!!!
Com Anita e Guto

### algoritmo 14

```
prog imp4
    imprima "Aprendendo Algoritmo !!!\n ";
    imprima "Com Anita e Guto";
fimprog
```

Vídeo

Aprendendo Algoritmo!!!
Com Anita e Guto

Observe que o símbolo `\n` poderá ser colocado ao final da linha anterior ou no início da próxima linha que produzirá o mesmo efeito mas, lembre-se, *sempre entre aspas*.

### algoritmo 15

```
prog imp5
    imprima "Aprendendo Algoritmo \n Com Anita e Guto\n\n E implementando no UAL\n
    Fica muito mais fácil!! "; # digite tudo na mesma linha
fimprog
```

↳ **Quando aparecer comentários do tipo:** `# continuacao` (ou `# continuacao da linha anterior`, significa que você deverá digitar tudo em uma só linha. **Não pressione enter**).

### Vídeo

Aprendendo Algoritmo

Com Anita e Guto

E implementando no UAL

Fica tudo mais fácil

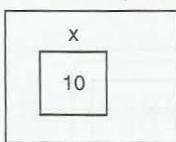
Observe que podemos, usando um único comando **imprima** e fazendo uso do símbolo `\n`, mostrar várias mensagens em várias linhas, inclusive deixando linha em branco quando colocamos `\n\n`.

### algoritmo 16

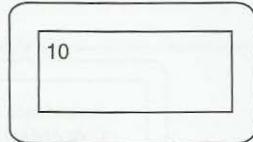
```
prog imp6
    int x;
    x <- 10;
    imprima  x ;
fimprog
```

Esse trecho é de muita importância pois x recebe o valor 10. Como já vimos, na memória do computador, existe uma variável chamada x e o valor 10 seria armazenado dentro da variável. Quando o comando **imprima** é executado, o valor de x, que está na memória do computador, é exibido pelo comando **imprima** no vídeo.

Memória Principal (MP)



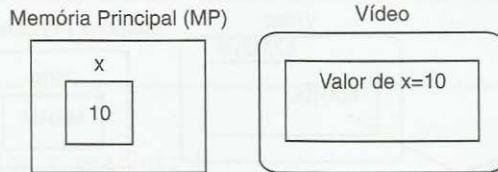
Vídeo



### algoritmo 17

```
prog imp7
    int x;
    x <- 10;
    imprima "Valor de x = ", x;
fimprog
```

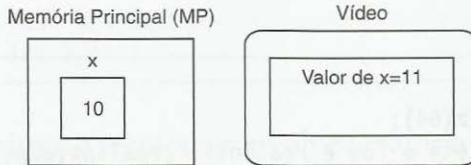
Esse trecho permite a exibição de uma mensagem e do conteúdo de uma variável na tela do computador.



### algoritmo 18

```
prog imp8
int x;
x <- 10;
imprima "Valor de x = ", x+1;
fimprog
```

Esse trecho é bem parecido com o anterior. O conteúdo da variável x é copiado da memória e acrescido de um, sendo impresso, após a string, sem alterar o valor de x na MP.



## COMANDO DE ENTRADA

É o comando que permite que o usuário digite dados, possibilitando um “diálogo com o computador”. O dado digitado é armazenado temporariamente em um registrador e, depois, copiado para a posição de memória indicada no comando. Lembre-se de que o nome de uma variável representa uma posição de memória.

Sintaxe:

leia	nome de uma variável	;
------	----------------------	---

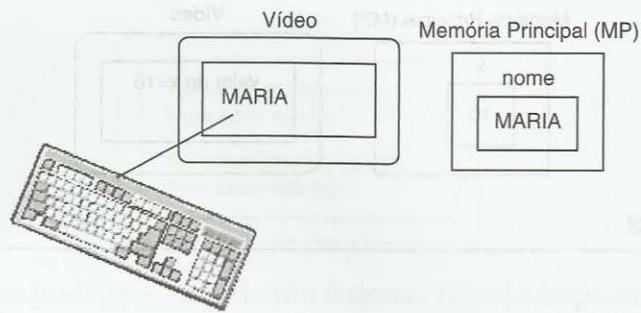
Exemplo 1: leia nome;

Como se lê?

Leia um valor para a variável nome.

O que faz o computador?

O computador fica “esperando” o usuário digitar um dado; neste exemplo, um nome: Maria. A variável nome, tendo sido declarada como string, recebe o valor Maria. Para facilitar, o dado digitado é sempre mostrado na tela.



⇨ Veja considerações sobre o comando de entrada para o interpretador sugerido no Apêndice I.

Observe o algoritmo a seguir que envolve a maioria das funções numéricas:

### algoritmo 19

```

prog teste
imprima "raiz: ",raiz(64);
imprima "\nraiz com exp e log e realint: ",realint(exp(1/2*log(64)));
imprima "\nraiz com exp e log sem realint: ",exp(1/2*log(64));
imprima "\n", formatar(sen(45*pi/180)+0.0001,3);
imprima "\npotencia com exp e log e formatar: ",formatar(exp(
    3*log(8))+0.001,3);#continuacao
imprima "\npotencia com exp e log sem formatar: ",exp(3*log(8));
imprima "\npotencia com operador ** e formatar: ",formatar(8**3,3);
imprima "\nraiz cubica: ",exp(1/3*log(8));
imprima "\nabsoluto: ",abs(-8);
imprima "\nabsoluto: ",abs(8);
imprima "\nconvertendo para inteiro 5.5: ",realint(5.5);
imprima "\nconvertendo para inteiro 6.5: ",realint(6.5);
imprima "\nconvertendo para inteiro 6.5 + 0.0001: ",realint(6.5+0.0001);
imprima "\nconvertendo para inteiro 5.45: ",realint(5.45);
imprima "\nconvertendo para inteiro 5.51: ",realint(5.51);
imprima "\nconvertendo para real 87: ",intreal(87);
imprima "\nconvertendo para int 3/4: ",realint(3/4),"\n";
fimprog

```

## VÍDEO

```
raiz: 8.0
raiz com exp e log e realint: 8
raiz com exp e log sem realint: 7.99999999999998
0.707
potencia com exp e log e formatar: 512.000
potencia com exp e log sem formatar: 511.999999999995
potencia com operador ** e formatar: 512.000
raiz cubica: 1.999999999999998
absoluto: 8
absoluto: 8
convertendo para inteiro 5.5: 6
convertendo para inteiro 6.5: 6
convertendo para inteiro 6.5 + 0.0001: 7
convertendo para inteiro 5.45: 5
convertendo para inteiro 5.51: 6
convertendo para real 87: 87.0
convertendo para int 3/4: 1
```

Observe o algoritmo a seguir que envolve a maioria das funções strings:

### algoritmo 20

```
prog funcoes
    string c,c1,d,d1;
    mprime "\ndigite palavra 1: ";
    leia c;
    imprima "\ndigite palavra 2: ";
    leia c1;
    imprima "\n",strtam(c);
    imprima "\nconcatenando: ",strconcat(c,c1);
    d <- strcopia(c);
    imprima "\nE O CONTEUDO DE D: ",d;
    d1<-strconcat(c,c1);
    imprima "\nconcatenacao: ",d1;
    imprima "\nprimeiro caractere: ",strprim(c);
    imprima "\nultimo caractere: ",strult(c);
    imprima "\ntodos menos o primeiro: ",strresto(c);
    imprima "\no terceiro elemento: ",strelem(c,3); # sairá o 4º caractere
    imprima "\nos tres primeiros elementos: ",strnprim(c,3);
    imprima "\nos tres ultimos elementos: ",strnresto(c,3);
    imprima "\n";
fimprog
```

## VÍDEO

```
digite palavra:TESTANDO
digite palavra2:UAL
tamanho da 1 palavra: 8
concatenando sem armazenar:TESTANDOUAL
o conteudo de d(variavel que teve valor copiado de c: TESTANDO
o conteudo de e(variavel que teve valor atribuido de c: TESTANDO
concatenacao com armazenamento:TESTANDOUAL
primeiro caractere:T
ultimo caractere:L
todos menos o primeiro:ESTANDOUAL
o terceiro elemento:S
os tres primeiros elementos:TES
os tres ultimos elementos:UAL
```

## Testando Hierarquia

### algoritmo 21

```
prog teste
    imprima "\nTESTANDO HIERARQUIA\n";
    imprima "\n12 + 5 / 2 é igual a: ", 12 + 5 / 2;
    imprima "\nÉ DIFERENTE DE (12 + 5)/2 que é igual a: ", (12 + 5)/2, "
        logo / tem HIERARQUIA MAIOR do que + ou -";#continuacao
    imprima "\n\n64**1/4 é igual a: ",64**1/4 ;
    imprima "\nÉ DIFERENTE de 64**(1/4)que é igual a: ",64**(1/4)," logo **
        tem HIERARQUIA MAIOR do que * ou / " ;#continuacao
    imprima "\n\n3*7 % 5 é igual a: ", 3*7 % 5;
    imprima "\nÉ DIFERENTE de (3 * 7) % 5 que é igual a: ",(3 * 7) % 5, "
        logo % tem HIERARQUIA MAIOR do que * " ;#continuacao
    imprima "\n\n3 * 7 div 5 é igual a: ", 3*7 div 5;
    imprima "\nÉ IGUAL a (3 * 7) div 5 : ",(3 * 7) div 5," logo div tem a
        MESMA HIERARQUIA da * ou / "#continuacao
    imprima "\n";
fimprog
```

## VÍDEO

### TESTANDO HIERARQUIA

$12 + 5 / 2$  é igual a: 14.5

É DIFERENTE DE  $(12 + 5) / 2$ , que é igual a: 8.5 s; logo, / tem HIERARQUIA MAIOR do que + ou -

$64**1/4$  é igual a: 16.0

É DIFERENTE de  $64^{**}(1/4)$ ; que é igual a: 2.8284271247461903; logo \*\* tem HIERARQUIA MAIOR do que \* ou /

$3*7 \% 5$  é igual a: 6

É DIFERENTE de  $(3 * 7) \% 5$ , que é igual a: 1; logo, % tem HIERARQUIA MAIOR do que \*

$3 * 7 \text{ div } 5$  é igual a: 4

É IGUAL a  $(3 * 7) \text{ div } 5 : 4$ ; logo, div tem a MESMA HIERARQUIA de \* ou /

Uma operação de divisão não pode ser um dos operandos de uma expressão com %, pois o operador % tem hierarquia maior do que a divisão. Logo,  $6\%2$  é executado primeiro e, quando for feita a divisão, o divisor será zero (0). Coloque a operação de divisão entre parênteses para ser executada primeiro.

Um número real pode ser usado como operando de div, porém será feito arredondamento. Isso é *desaconselhável*, pois acaba com a filosofia de div.

### algoritmo 22

```
prog teste
imprima "\nTESTANDO HIERARQUIA";
imprima "\n\n18/6 % 2 é igual a: ", 18 / 6 % 2;
imprima "\nUma operação de divisão fora de parênteses não pode ser um
dos operandos de uma expressao com %.";#continuacao
imprima "\n\n20 / 4 div 2 é igual a: ", 20 / 4 div 2;
imprima "\nÉ IGUAL a (20 / 4) div 2 : ",(20 / 4) div 2," logo div tem a
MESMA HIERARQUIA "#continuacao
imprima " da /. ";
imprima "\n\n30 / 4 div 2 é igual a: ", 30/4 div 2;
imprima "\nÉ IGUAL a (30 / 4) div 2 : ",(30 / 4) div 2," logo div tem
a MESMA HIERARQUIA "#continuacao
imprima " da / ";
```

```

imprima "\n\n7. div 4: ",7. div 4;
imprima "\n7 div 4: ",7 div 4;
imprima "\n6. div 4: ",6. div 4;
imprima "\n6 div 4: ",6 div 4;
imprima "\n";
fimprog

```

## VÍDEO

### TESTANDO HIERARQUIA

$18/6 \% 2$  é igual a: Infinity

Uma operação de divisão fora de parênteses não pode ser um dos operandos de uma expressão com %.

$20 / 4 \text{ div } 2$  é igual a: 2

É IGUAL a  $(20 / 4) \text{ div } 2 : 2$ ; logo, div tem a MESMA HIERARQUIA da /.

$30 / 4 \text{ div } 2$  é igual a: 4

É IGUAL a  $(30 / 4) \text{ div } 2 : 4$ ; logo, div tem a MESMA HIERARQUIA da /

$7. \text{ div } 4: 2$

$7 \text{ div } 4: 1$

$6. \text{ div } 4: 2$

$6 \text{ div } 4: 1$

Muita atenção para o uso dos operadores % e div numa expressão.

Se % vier antes, não tem problema.

### algoritmo 23

Entrar com um número inteiro de 3 casas e imprimir o algarismo da casa das dezenas.

```

prog teste
int a,d;
imprima "\nDigite numero de tres casas: ";
leia a;
d<-a % 100 div 10;
imprima "\nAlgarismo da casa das dezenas: ",d;
imprima "\n";
fimprog

```

## VÍDEO

Digite numero de tres casas:135

algarismo da casa das dezenas: 3

Se a idéia é efetuar a operação com div antes, teremos problema, pois o operador % tem hierarquia maior que div; logo, primeiro será operado  $10 \% 10$ , cujo o resultado é zero (0). Isso impossibilita a operação com div, uma vez que não se pode dividir por 0.

### algoritmo 24

```
prog teste
    int a,d;
    imprima "\nDigite numero de tres casas: ";
    leia a;
    d<-a div 10 % 10;
    imprima "\nalgarismo da casa das dezenas: ", d;
    imprima "\n";
fimprog
```

## VÍDEO

Digite numero de tres casas:135

Floating point exception (core dumped)

Lembre-se sempre disto: quando você montar uma expressão com div e %, se div vier antes de %, coloque parênteses para priorizar uma operação de hierarquia menor.

### algoritmo 24

Sem problema

```
prog teste
    int a,d;
    imprima "\nDigite numero de tres casas: ";
    leia a;
    d<-(a div 10) % 10;
    imprima "\nalgarismo da casa das dezenas: ",d;
    imprima "\n";
fimprog
```

## VÍDEO

Digite numero de tres casas:135

algarismo da casa das dezenas: 3

## Algumas aplicações com % e div

### algoritmo 25

*Entrar com uma data no formato ddmmaa e imprimir: dia, mês e ano separados.*

```
prog teste
int data,dia,mes,ano;
imprima "\nDigite data no formato ddmmaa: ";
leia data;
dia<-data div 10000;
mes<-data % 10000 div 100;
ano<-data %100;
imprima "\nDIA: ",dia;
imprima "\nMES: ",mes;
imprima "\nANO: ",ano;
imprima "\n";
fimprog
```

## VÍDEO

Digite data no formato DDMMAA:230389

DIA:23

MES:3

ANO:89

### algoritmo 26

*Entrar com uma data no formato ddmmaa e imprimir no formato mmddaa.*

```
prog teste
int data,dia,mes,ano,ndata;
imprima "\nDigite data no formato DDMMAA: ";
leia data;
dia<-data div 10000;
mes<-data % 10000 div 100;
```

```
ano<-data %100;  
ndata<-mes *10000 +dia*100+ano;  
imprima "\nDIA: ",dia;  
imprima "\nMES: ",mes;  
imprima "\nANO: ",ano;  
imprima "\n\nDATA NO FORMATO MMDDAA: ",ndata;  
imprima "\n";  
fimprog
```

### VÍDEO

Digite data no formato DDMMAA:251201

DIA:25  
MES:12  
ANO:1

DATA NO FORMATO MMDDAA:122501

Os operadores ++ e --

### algoritmo 27

```
prog xx  
int x,y;  
x<-2;  
y<-5;  
imprima "\nx = ",x;  
x++;  
imprima "\ny = ",y;  
y--;  
imprima "\nnovo valor de x = ",x;  
imprima "\nnovo valor de y = ",y;  
imprima "\n\n";  
fimprog
```

### VÍDEO

x = 2

y = 5

novo valor de x = 3

novo valor de y = 4

## Só reforçando

- Todas as palavras reservadas são escritas com letras minúsculas.
- O operador de atribuição deverá ser formado pelo sinal < seguido do sinal -, ficando: <- .
- Os identificadores (nome do algoritmo e das variáveis deverão começar por uma letra, e os demais caracteres por letra ou algarismo).
- Os comandos: imprima, leia, atribuição e as declarações de variáveis terminam com ; .
- Os comandos prog e fimprog não têm ; .

## EXERCÍCIOS – LISTA 1 LEIA, IMPRIMA, ATRIBUIÇÃO E FUNÇÕES

### algoritmo 28

*Imprimir a mensagem: "É PRECISO FAZER TODOS OS ALGORITMOS PARA APRENDER".*

```
prog lea1
    imprima "\nE PRECISO FAZER TODOS OS ALGORITMOS PARA APRENDER ";
    imprima "\n";
fimprog
```

↳ **imprima "\n";** será sempre colocada para que o prompt não fique na mesma linha da última impressão, uma vez que o cursor não desce automaticamente.

### algoritmo 29

*Imprimir seu nome.*

```
prog lea2
    imprima "\n <seu nome>";
    imprima "\n";
fimprog
```

### algoritmo 30

*Criar um algoritmo que imprima o produto entre 28 e 43.*

```
prog lea3
    int produto;
    produto <- 28 * 43;
    imprima "\nO produto entre os dois é: ", produto;
    imprima "\n";
fimprog
```

## algoritmo 31

85 omniphis

Criar um algoritmo que imprima a média aritmética entre os números 8, 9 e 7.

```
prog lea4
    real ma;
    ma <- ( 8 + 9 + 7 ) / 3;
    imprima "\nA media aritmetica e: ", ma;
    imprima "\n";
fimprog
```

## algoritmo 32

Ler um número inteiro e imprimi-lo.

```
prog lea5
    int num;
    imprima "\n entre com um numero: ";
    leia num;
    imprima "\nnumero : ", num;
    imprima "\n";
fimprog
```

## algoritmo 33

Ler dois números inteiros e imprimi-los.

```
prog lea6
    int num1, num2;
    imprima "\n entre com um numero: ";
    leia num1;
    imprima "\n entre com outro numero: ";
    leia num2;
    imprima "\nnumero 1 : ", num1;
    imprima "\nnumero 2 : ", num2;
    imprima "\n";
fimprog
```

## algoritmo 34

Ler um número inteiro e imprimir seu sucessor e seu antecessor.

```
prog lea7
    int numero, suc, ant;
    imprima "\n entre com um numero: ";
    leia numero;
    ant <- numero -1;
    suc <- numero +1;
    imprima "\no sucessor e b", suc,"b o antecessor e b", ant;
    imprima "\n";
fimprog
```

## algoritmo 35

---

Ler nome, endereço e telefone e imprimir-los.

```
prog lea8
    string nome, endereco, telefone;
    imprima "\nentre com nome: ";
    leia nome;
    imprima "\nentre com endereco: ";
    leia endereco;
    imprima "\nentre com telefone: ";
    leia telefone;
    imprima "\n\n\n";
    imprima "\nNome : ", nome;
    imprima "\nEndereco: ", endereco;
    imprima "\nTelefone: ", telefone;
    imprima "\n";
fimprog
```

## algoritmo 36

---

Ler dois números inteiros e imprimir a soma. Antes do resultado, deverá aparecer a mensagem: Soma.

```
prog lea9
    int num1, num2, soma;
    imprima "\n entre com um numero: ";
    leia num1;
    imprima "\n entre com outro numero: ";
    leia num2;
    soma <- num1 + num2;
    imprima "\nSoma: ", soma;
    imprima "\n";
fimprog
```

## algoritmo 37

---

Ler dois números inteiros e imprimir o produto.

```
prog lea10
    int num1, num2, prod;
    imprima "\n entre com um numero: ";
    leia num1;
    imprima "\n entre com outro numero: ";
    leia num2;
    prod <- num1 * num2;
    imprima "\nproduto: ", prod;
    imprima "\n";
fimprog
```

## algoritmo 38

---

Ler um número real e imprimir a terça parte deste número.

```
prog lea11
    real num;
    imprima "\nentre com um numero com ponto: ";
    leia num;
    imprima "\na terça parte e: ", num/3;
    imprima "\n";
fimprog
```

## algoritmo 39

---

Entrar com dois números reais e imprimir a média aritmética com a mensagem "média" antes do resultado.

```
prog lea12
    real nota1, nota2, media;
    imprima "\ndigite 1a nota: ";
    leia nota1;
    imprima "\ndigite 2a nota: ";
    leia nota2;
    media <- ( nota1 + nota2)/2;
    imprima "\nmedia: ", media;
    imprima "\n";
fimprog
```

## algoritmo 40

---

Entrar com dois números inteiros e imprimir a seguinte saída:

```
dividendo:
divisor:
quociente:
resto:
prog lea13
    int quoc, rest, val1, val2;
    imprima "\nentre com o dividendo: ";
    leia val1;
    imprima "\nentre com divisor: ";
    leia val2;
    quoc <- val1 div val2;
    rest <- val1 % val2;
    imprima "\n\n\n";
    imprima "\ndividendo : ", val1;
    imprima "\ndivisor : ", val2;
    imprima "\nquociente : ", quoc;
    imprima "\nresto : ", rest;
    imprima "\n";
fimprog
```

## algoritmo 41

SE omniphys

Entrar com quatro números e imprimir a média ponderada, sabendo-se que os pesos são respectivamente: 1, 2, 3 e 4.

```
prog lea14
    real a, b, c, d, mp;
    imprima "\nentre com 1 numero: ";
    leia a;
    imprima "\nentre com 2 numero: ";
    leia b;
    imprima "\nentre com 3 numero: ";
    leia c;
    imprima "\nentre com 4 numero: ";
    leia d;
    mp <- (a*1 + b*2 + c*3 + d*4)/10;
    imprima "\nmedia ponderada: ", mp;
    imprima "\n";
fimprog
```

## algoritmo 42

Entrar com um ângulo em graus e imprimir: seno, co-seno, tangente, secante, co-secante e co-tangente deste ângulo.

```
prog lea15
    real angulo, rang;
    imprima "\ndigite um angulo em graus: ";
    leia angulo;
    rang <- angulo*pi/180;
    imprima "\nseno: ", sen(rang);
    imprima "\nco-seno: ", cos(rang);
    imprima "\ntangente: ", tan(rang);
    imprima "\nco-secante: ", 1/sen(rang);
    imprima "\nsecante: ", 1/cos(rang);
    imprima "\ncotangente: ", 1/tan(rang);
    imprima "\n";
fimprog
```

↳ Alguns ângulos que você digitar poderão não ter resposta em algumas funções, mas este problema será resolvido quando você aprender a estrutura de teste.

## algoritmo 43

Entrar com um número e imprimir o logaritmo desse número na base 10.

```
prog lea16
    real num, logaritmo;
```

```
imprima "\nentre com o logaritmando: ";
leia num;
logaritmo <- log(num) / log(10);
imprima "\nlogaritmo: ", logaritmo;
imprima "\n";
fimprog
```

### algoritmo 44

Entrar com o número e a base em que se deseja calcular o logaritmo desse número e imprimi-lo.

```
prog lea17
real num, base, logaritmo;
imprima "\nentre com o logaritmando: ";
leia num;
imprima "\nentre com a base: ";
leia base;
logaritmo <- log(num) / log(base);
imprima "\no logaritmo de", num, "é na base", base, "é:", logaritmo;
imprima "\n";
fimprog
```

### algoritmo 45

Entrar com um número e imprimir a seguinte saída:

```
numero:
quadrado:
raiz quadrada:
prog lea18
real num, quad, raizquad;
imprima "\ndigite numero: ";
leia num;
quad <- num ** 2;
raizquad <- raiz(num);
imprima "\nnumero: ", num;
imprima "\nquadrado: ", quad;
imprima "\nraiz quadrada: ", raizquad;
imprima "\n";
fimprog
```

### algoritmo 46

Fazer um algoritmo que possa entrar com o saldo de uma aplicação e imprima o novo saldo, considerando o reajuste de 1%.

```
prog lea19
real saldo, nsaldo;
imprima "\ndigite saldo: "; n/" .0000 ";
nsaldo <- saldo * 1.01;
```

```
leia saldo;
nsaldo <- saldo * 1.01;
imprima "\nnovo saldo: ", nsaldo;
imprima "\n";
fimprog
```

### algoritmo 47

---

*Entrar com um número no formato CDU e imprimir invertido: UDC. (Exemplo: 123, sairá 321.) O número deverá ser armazenado em outra variável antes de ser impresso.*

```
prog lea20
int num, c, d, u, num1;
imprima "\nentre com um número de 3 dígitos: ";
leia num;
c <- num div 100;
d <- num % 100 div 10;
u <- num % 10;
num1 <- u*100 + d*10 + c;
imprima "\nnúmero: ", num;
imprima "\ninvertido: ", num1;
imprima "\n";
fimprog
```

### algoritmo 48

---

*Antes de o racionamento de energia ser decretado, quase ninguém falava em quilowatts; mas, agora, todos incorporaram essa palavra em seu vocabulário. Sabendo-se que 100 quilowatts de energia custa um sétimo do salário mínimo, fazer um algoritmo que receba o valor do salário mínimo e a quantidade de quilowatts gasta por uma residência e calcule. Imprima:*

- o valor em reais de cada quilowatt
- o valor em reais a ser pago
- o novo valor a ser pago por essa residência com um desconto de 10%.

```
prog lea21
real sm, qtdade, preco, vp, vd;
imprima "\nentre com o salário mínimo: ";
leia sm;
imprima "\nentre com a quantidade em quilowatt: ";
leia qtdade;
# divide por 7 para achar o preço de 100 Kw e por 100 para achar de 1 Kw
preco <- sm /700;
vp <- preco * qtdade;
vd <- vp * 0.9;
imprima "\npreço do quilowatt: ", preco, "\n valor a ser pago: ", vp,
```

```
"\n valor com desconto: ", vd;  
imprima "\n";  
fimprog
```

## algoritmo 49

---

*Entrar com um nome e imprimir:*

```
todo nome:  
primeiro caractere:  
ultimo caractere:  
do primeiro ate o terceiro:  
quarto caractere:  
todos menos o primeiro:  
os dois ultimos:  
prog lea22  
string nome;  
int n;  
imprima "\nentre com nome: ";  
leia nome;  
imprima "\ntodo nome: " , nome;  
imprima "\nprimeiro caractere: " , strprim(nome);  
imprima "\nultimo caractere: " , strult(nome);  
imprima "\nprimeiro ao terceiro caractere: " , strnprim(nome,3);  
imprima "\nquarto caractere: " , strelem(nome,3);  
imprima "\ntodos menos o primeiro: " , strresto ( nome);  
n <-strtam(nome) -2;  
imprima "\nos dois ultimos: " , strnresto(nome,n);  
imprima "\n";  
fimprog
```

## algoritmo 50

---

*Entrar com a base e a altura de um retângulo e imprimir a seguinte saída:*

```
perimetro:  
area:  
diagonal:  
prog lea23  
real perimetro, area, diagonal, base, altura;  
imprima "\ndigite base: ";  
leia base;  
imprima "\ndigite altura: ";  
leia altura;  
perimetro <- 2*(base + altura);  
area <-base * altura;  
diagonal <- raiz(base**2 + altura**2);  
imprima "\nperimetro = " ,perimetro;  
imprima "\narea = " , area ;
```

```
imprima "\ndiagonal = ", diagonal ;
imprima "\n";
fimprog
```

## algoritmo 51

*Entrar com o raio de um círculo e imprimir a seguinte saída:*

```
perimetro:
area:
prog lea24
real raio, perimetro, area;
imprima "\ndigite raio: ";
leia raio;
perimetro <- 2* pi * raio;
area <- pi *raio ** 2;
imprima "\nperimetro : ", perimetro;
imprima "\narea : ", area ;
imprima "\n";
fimprog
```

## algoritmo 52

*Entrar com o lado de um quadrado e imprimir:*

```
perimetro:
area:
diagonal:
prog lea25
real lado, perimetro, area, diagonal;
imprima "\ndigite o lado do quadrado: ";
leia lado ;
perimetro <- 4 * lado;
area<- lado ** 2;
diagonal <- lado * raiz(2);
imprima "\nperimetro: ", perimetro;
imprima "\narea: ", area;
imprima "\ndiagonal: ", diagonal;
imprima "\n";
fimprog
```

## algoritmo 53

*Entrar com os lados a, b, c de um paralelepípedo. Calcular e imprimir a diagonal.*

```
prog lea26
real a, b, c, diagonal;
imprima "\nentre com a base: ";
leia a;
imprima "\nentre com a altura: ";
leia b;
```

```

imprima "\nentre com a profundidade: ";
leia c;
diagonal <-raiz( a**2 + b**2 + c**2 );
imprima "\ndiagonal : ", diagonal;
imprima "\n";
fimprog

```

### algoritmo 54

*Criar um algoritmo que calcule e imprima a área de um triângulo.*

```
prog lea27
    real a, b;
    imprima "\nEntre com a base: ";
    leia a;
    imprima "\nEntre a altura do um
    leia b;
    imprima "\nArea = ", (a * b)/2;
    imprima "\n";
fimprog
```

### **algoritmo 55**

*Criar um algoritmo que calcule e imprima a área de um losango.*

```

prog lea28
real diagmaior, diagmenor, area;
imprima "\nmedida da diagonal maior: ";
leia diagmaior;
imprima "\nmedida da diagonal menor: ";
leia diagmenor;
area <- (diagmaior * diagmenor)/2;
imprima "\narea =", area;
imprima "\n";
fimprog

```

### algoritmo 56

*Entrar com nome e idade. Imprimir a seguinte saída:*

```
nome:      S = shabnam's album
idade:     S = shabnam's album
S = shabnam's album
S = shabnam's album
prog lea29
string nome;
int idade;
imprima "\ndigite nome: ";
leia nome;
imprima "\ndigite idade: ";
leia idade;
# a linha abaixo é para dar uma separação entre a entrada e a saída
```

```

imprima "\n\n";
imprima "\nnome = ", nome;
imprima "\nidade = ", idade;
imprima "\n";
fimprog

```

## algoritmo 57

---

*Entrar com as notas da PR1 e PR2 e imprimir a média final:*

- truncada:
- arredondada:

```

prog lea30
real pr1, pr2, mf;
imprima "\ndigite pr1: ";
leia pr1;
imprima "\ndigite pr2: ";
leia pr2;
mf <- ( pr1 + pr2 ) / 2;
imprima "\nmedia truncada = ", realint((mf- 0.5)+0.001);
imprima "\nmedia arredondada = ", realint( mf+0.001);
imprima "\n";
fimprog

```

### VÍDEO

digite pr1:2.3	digite pr1:7.9
digite pr2:3.7	digite pr2:8.1
media truncada = 3	media truncada = 8
media arredondada = 3	media arredondada = 8
digite pr1:2.8	digite pr1:6.9
digite pr2:2.7	digite pr2:8.1
media truncada = 2	media truncada = 7
media arredondada = 3	media arredondada = 8

## algoritmo 58

---

*Entrar com valores para xnum1, xnum2 e xnum3 e imprimir o valor de x, sabendo-se que:*

$$X = xnum1 + \frac{xnum2}{xnum3 + xnum1} + 2(xnum1 - xnum2) + \log_2^{64}$$

```
prog lea31
real xnum1, xnum2, xnum3, x;
imprima "\nEntrar com 1 valor: ";
leia xnum1;
imprima "\nEntrar com 2 valor: ";
leia xnum2;
imprima "\nEntrar com 3 valor: ";
leia xnum3;
x <- xnum1 + xnum2 / (xnum3 + xnum1) + 2 * (xnum1 - xnum2) + log(64.)/
log(2.);
imprima "\nX = ", x;
imprima "\n";
fimprog
```

## algoritmo 59

---

*Entrar com os valores dos catetos de um triângulo retângulo e imprimir a hipotenusa.*

```
prog lea32
real a,b,c;
imprima "\nEntrar com 1 cateto: ";
leia b;
imprima "\nEntrar com 2 cateto: ";
leia c;
a <- raiz (b**2 + c**2);
imprima "\nA hipotenusa e: ", a;
imprima "\n";
fimprog
```

## algoritmo 60

---

*Entrar com a razão de uma PA e o valor do 1º termo. Calcular e imprimir o 10º termo da série.*

```
prog lea33
int dec, razao, termo;
imprima "\nEntrar com o 1º termo: ";
leia termo;
imprima "\nEntrar com a razao: ";
leia razao;
dec <- termo + 9* razao;
imprima "\nO 10º termo desta P.A. e: ", dec;
imprima "\n";
fimprog
```

## algoritmo 61

---

Entrar com a razão de uma PG e o valor do 1º termo. Calcular e imprimir o 5º termo da série.

```
prog lea34
    int quinto, razao, termo;
    imprima "\nEntre com o 1º termo: ";
    leia termo;
    imprima "\nEntre com a razao: ";
    leia razao;
    quinto <- termo * razao ^4;
    imprima "\nO 5º termo desta P.G. é: ", quinto;
    imprima "\n";
fimprog
```

## algoritmo 62

---

Em épocas de pouco dinheiro, os comerciantes estão procurando aumentar suas vendas oferecendo desconto. Faça um algoritmo que possa entrar com o valor de um produto e imprima o novo valor tendo em vista que o desconto foi de 9%.

```
prog lea35
    real preco, npreco;
    imprima "\ndigite valor do produto: ";
    leia preco;
    npreco <- preco * 0.91;
    imprima "\npreco com desconto: ", npreco;
    imprima "\n";
fimprog
```

## algoritmo 63

---

Criar um algoritmo que efetue o cálculo do salário líquido de um professor. Os dados fornecidos serão: valor da hora aula, número de aulas dadas no mês e percentual de desconto do INSS.

```
prog lea36
    int na;
    real vha, pd, td, sb, sl;
    imprima "\nhoras trabalhadas: ";
    leia na ;
    imprima "\nvalor da hora-aula: ";
    leia vha ;
    imprima "\npercentual de desconto: ";
    leia pd ;
    sb <- na * vha;
    td <- (pd / 100) * sb;
    sl <- sb - td;
```

```
imprima "\nsalario liquido: ",sl;
imprima "\n";
fimprog
```

## algoritmo 64

---

Ler uma temperatura em graus centígrados e apresentá-la convertida em graus Fahrenheit. A fórmula de conversão é:  $F = \frac{9.c + 160}{5}$  onde F é a temperatura em Fahrenheit e C é a temperatura em centígrados.

```
prog lea37
real f, c;
imprima "\ndigite o valor da temperatura em graus centigrados: ";
leia c;
f <- ( 9 * c + 160)/5;
imprima "\no valor da temperatura em graus fahrenheit e =", f;
imprima "\n";
fimprog
```

## algoritmo 65

---

Calcular e apresentar o valor do volume de uma lata de óleo, utilizando a fórmula:  $volume = 3.14159 * R^2 * altura$ .

```
prog lea38
real volume, altura, raio;
imprima "\ndigite a altura da lata: ";
leia altura;
imprima "\ndigite o raio da lata: ";
leia raio;
volume <- pi *raio ** 2 *altura;
imprima "\no volume da lata e =", volume;
imprima "\n";
fimprog
```

## algoritmo 66

---

Efetuar o cálculo da quantidade de litros de combustível gastos em uma viagem, sabendo-se que o carro faz 12 km com um litro. Deverão ser fornecidos o tempo gasto na viagem e a velocidade média.

Utilizar as seguintes fórmulas:

$$distância = tempo \times velocidade.$$

$$litros\ usados = distância / 12.$$

O algoritmo deverá apresentar os valores da velocidade média, tempo gasto na viagem, distância percorrida e a quantidade de litros utilizados na viagem.

```

prog lea39
    real tempo, vel, dist, litros;
    imprima "\ndigite o tempo gasto: ";
    leia tempo;
    imprima "\ndigite a velocidade media: ";
    leia vel;
    dist <- tempo * vel;
    litros <- dist / 12;
    imprima "\nvelocidade = ", vel, "\ntempo = ", tempo, "\ndistancia = ",
    dist, "\nlitros = ", litros;
    imprima "\n";
fimprog

```

### **algoritmo 67**

---

*Efetuar o cálculo do valor de uma prestação em atraso, utilizando a fórmula:  
prestação = valor + (valor\*(taxa/100)\*tempo).*

```

prog lea40
    real prest, valor, taxa;
    int tempo;
    imprima "\ndigite o valor da prestação: ";
    leia valor;
    imprima "\ndigite a taxa: ";
    leia taxa;
    imprima "\ndigite o tempo(numero de meses): ";
    leia tempo;
    prest <- valor+(valor*(taxa/100)*tempo);
    imprima "\no valor da prestacao em atraso e =", prest;
    imprima "\n";
fimprog

```

### **algoritmo 68**

---

*Ler dois valores para as variáveis A e B, efetuar a troca dos valores de forma que a variável A passe a ter o valor da variável B e que a variável B passe a ter o valor da variável A. Apresentar os valores trocados.*

```

prog lea41
    real a, b, aux;
    imprima "\ndigite 1 numero com ponto: ";
    leia a;
    imprima "\ndigite 2 numero com ponto: ";
    leia b;
    aux <- a;
    a <- b;
    b <- aux;
    imprima "\na = ", a, "\nb = ", b;

```

```
imprima "\n";
fimprog
```

## algoritmo 69

---

Criar um algoritmo que leia o numerador e o denominador de uma fração e transformá-lo em um número decimal.

```
prog lea42
int num, denom;
imprima "\ndigite numerador: ";
leia num;
imprima "\ndigite denominador: ";
leia denom;
imprima "\ndecimal: ", num / denom;
imprima "\n";
fimprog
```

## algoritmo 70

---

Todo restaurante, embora por lei não possa obrigar o cliente a pagar, cobra 10% para o garçom. Fazer um algoritmo que leia o valor gasto com despesas realizadas em um restaurante e imprima o valor total com a gorjeta.

```
prog lea43
real cres, cgorj;
imprima "\nEnter com o valor da conta: ";
leia cres;
cgorj <- cres *1.1;
imprima "\nO valor da conta com a gorjeta sera: ", formatar(cgorj,2);
imprima "\n";
fimprog
```

## algoritmo 71

---

Criar um algoritmo que leia um valor de hora e informe quantos minutos se passaram desde o início do dia.

```
prog lea44
int hora, tminuto, minuto;
imprima "\nentre com hora atual: ";
leia hora;
imprima "\nentre com minutos: ";
leia minuto;
tminuto <- hora * 60 + minuto;
imprima "\nAte agora se passaram: ", tminuto, " minutos";
imprima "\n";
fimprog
```

## algoritmo 72

---

Criar um algoritmo que leia o valor de um depósito e o valor da taxa de juros. Calcular e imprimir o valor do rendimento e o valor total depois do rendimento.

```
prog lea45
    real deposito, taxa, valor, total;
    imprima "\nentre com depósito: ";
    leia deposito;
    imprima "\nentre coma taxa de juros: ";
    leia taxa;
    valor <- deposito*taxa/100;
    total <- deposito + valor;
    imprima "\nRendimentos: ", valor, "\nTotal: ", total;
    imprima "\n";
fimprog
```

## algoritmo 73

---

Criar um algoritmo que receba um número real, calcular e imprimir:

- a parte inteira do número
- a parte fracionária do número
- o número arredondado

```
prog lea46
    real num, numfrac;
    int numi, numa;
    imprima "\nentre com um numero com parte fracionaria: ";
    leia num;
    numi <- realint((num - 0.5));
    numfrac <- num - numi;
    numa <- realint(num + 0.00001);
    imprima "\nparte inteira: ", numi;
    imprima "\nparte fracionaria: ", formatar((numfrac + 0.00001),3);
    imprima "\nnúmero arredondado: ", numa;
    imprima "\n";
fimprog
```

---

↳ Qualquer dúvida, consulte o Apêndice I.

---

## VÍDEO

entre com um numero com parte fracionaria:7.1

parte inteira:7

parte fracionária:0.100  
numero arredondado:7

entre com um numero com parte fracionaria:8.5

parte inteira:8

parte fracionária:0.500  
numero arredondado:9

entre com um numero com parte fracionaria:7.49

parte inteira:7

parte fracionária:0.490  
numero arredondado:7

entre com um numero com parte fracionaria:8.4999

parte inteira:8

parte fracionária:0.499  
numero arredondado:8

entre com um numero com parte fracionaria:7.4999

parte inteira:7

parte fracionária:0.499  
numero arredondado:7

entre com um numero com parte fracionária:8.0

parte inteira:8

parte fracionária:1.000  
numero arredondado:8

## algoritmo 74

Para vários tributos, a base de cálculo é o salário mínimo. Fazer um algoritmo que leia o valor do salário mínimo e o valor do salário de uma pessoa. Calcular e imprimir quantos salários mínimos ela ganha.

```
prog lea47
real salmin, salpe, num;
imprima "\nentre com o salario minimo: ";
leia salmin;
imprima "\nentre com o salario da pessoa: ";
leia salpe;
num <- salpe / salmin;
imprima "\na pessoa ganha ", num, " salarios minimos";
imprima "\n";
fimprog
```

## algoritmo 75

Criar um algoritmo que leia o peso de uma pessoa, só a parte inteira, calcular e imprimir:

- o peso da pessoa em gramas
- novo peso, em gramas, se a pessoa engordar 12%

```
prog lea48
    int peso, pesogramas, novopeso;
    imprima "\nentre com seu peso, só a parte inteira: ";
    leia peso;
    pesogramas <- peso * 1000;
    novopeso <- pesogramas * 1.12;
    imprima "\npeso em gramas: ", pesogramas;
    imprima "\nnovo peso: ", novopeso;
    imprima "\n";
fimprog
```

### algoritmo 76

---

Criar um algoritmo que leia um número entre 0 e 60 e imprimir o seu sucessor, sabendo que o sucessor de 60 é 0. Não pode ser utilizado nenhum comando de seleção e nem de repetição.

```
prog leia49
    int num;
    imprima "\ndigite numero: ";
    leia num;
    imprima "\nsucessor: ", (num + 1) % 61;
    imprima "\n";
fimprog
```

### algoritmo 77

---

Ler dois números reais e imprimir o quadrado da diferença do primeiro valor pelo segundo e a diferença dos quadrados.

```
prog lea50
    real a, b, d, q;
    imprima "\ndigite 1 numero: ";
    leia a;
    imprima "\ndigite 2 numero: ";
    leia b;
    d <- (a - b)**2;
    q <- a**2 - b**2;
    imprima "\no quadrado da diferenca =", d , "\ndiferenca dos quadrados =", q;
    imprima "\n";
fimprog
```

### algoritmo 78

---

Dado um polígono convexo de n lados, podemos calcular o número de diagonais diferentes ( $nd$ ) desse polígono pela fórmula :  $nd = n(n - 3) / 2$ . Fazer um algoritmo que leia quantos lados tem o polígono, calcule e escreva o número de diagonais diferentes ( $nd$ ) do mesmo.

```

prog lea51
real nd;
int n;
imprima "\ndigite o numero de lados do poligono: ";
leia n;
nd <- n * ( n - 3 ) / 2;
imprima "\nnumero de diagonais: ", nd;
imprima "\n";
fimprog

```

## algoritmo 79

---

*Uma pessoa resolveu fazer uma aplicação em uma poupança programada. Para calcular seu rendimento, ela deverá fornecer o valor constante da aplicação mensal, a taxa e o número de meses. Sabendo-se que a fórmula usada para este cálculo é:*

$$\text{valor acumulado} = P * \frac{(1+i)^n - 1}{i}$$

*i = taxa  
P = aplicação mensal  
n = número de meses*

```

prog lea52
real va, i, p;
int n;
imprima "\ndigite o valor da aplicacao: ";
leia p;
imprima "\ndigite a taxa( 0 - 1): ";
leia i;
imprima "\ndigite o numero de meses: ";
leia n;
va <- p*((1+i)**n)-1) / i;
imprima "\n0 valor acumulado: ", va;
imprima "\n";
fimprog

```

## algoritmo 80

---

*Criar um algoritmo que leia a quantidade de fitas que uma locadora de vídeo possui e o valor que ela cobra por cada aluguel, mostrando as informações pedidas a seguir:*

- Sabendo que um terço das fitas são alugadas por mês, exiba o faturamento anual da locadora;
- Quando o cliente atrasa a entrega, é cobrada uma multa de 10% sobre o valor do aluguel. Sabendo que um décimo das fitas alugadas no mês são devolvidas com atraso, calcule o valor ganho com multas por mês;
- Sabendo ainda que 2% de fitas se estragam ao longo do ano, e um décimo do total é comprado para reposição, exiba a quantidade de fitas que a locadora terá no final do ano.

```

prog lea53
int quant;
real valAluguel, fatAnual, multas, quantFinal;
imprima "\n Digite a quantidade de fitas: ";
leia quant;
imprima "\n Digite o valor do aluguel: ";
leia valAluguel;
fatAnual <- quant/3 * valAluguel * 12;
imprima "\n Faturamento anual: ", fatAnual;
multas <- valAluguel * 0.1 * (quant/3)/10;
imprima "\n Multas mensais: ", multas;
quantFinal <- quant - quant * 0.02 + quant/10; /* quant * 1.08 */
imprima "\n Quantidade de fitas no final do ano : ", quantFinal;
imprima "\n";
fimprog

```

## algoritmo 81

---

*Criar um algoritmo que, dado um número de conta corrente com três dígitos, retorne o seu dígito verificador, o qual é calculado da seguinte maneira:*

*Exemplo: número da conta: 235*

- Somar o número da conta com o seu inverso:  $235 + 532 = 767$
- multiplicar cada dígito pela sua ordem posicional e somar estes resultados:  $7 \cdot 1 + 6 \cdot 2 + 7 \cdot 3 = 40$

- o último dígito desse resultado é o dígito verificador da conta ( $40 \rightarrow 0$ ).

```

prog lea54
int conta, inv, digito, d1, d2, d3,soma;
imprima "\nDigite conta de tres digitos: ";
leia conta;
d1 <- conta div 100;
d2 <- conta % 100 div 10;
d3 <- conta % 100 % 10;
inv <- d3 *100 + d2 *10 +d1;
soma <- conta + inv;
d1 <- (soma div 100) * 1;
d2 <- (soma % 100 div 10) * 2;
d3 <- (soma % 100 % 10) *3;
digito <- (d1 +d2 +d3) % 10;
imprima "\ndigito verificador: ", digito;
imprima "\n";
fimprog

```

## ATENÇÃO

Refazer esta lista colocando todos os testes, usando o comando `se`, quando você aprender, nos exercícios necessários:

1. testar se o divisor é diferente de 0.
2. testar se o radicando é maior ou igual a 0.
3. testar se o logaritmando é maior do que 0 e a base, maior do que 0 e a base diferente de 1.
4. testar se o seno é diferente de zero quando se deseja calcular co-tangente e co-secante.
5. testar se o co-seno é diferente de zero quando se deseja calcular tangente e secante.
6. testar se os valores para lados de figuras são maiores do que zero.
7. e outros mais.



## Capítulo 3

# Estruturas de seleção

### CONCEITOS

Nossos algoritmos até agora seguiram um mesmo padrão: entrava-se com dados, estes eram processados e alguma informação era mostrada na tela.

Dessa forma, o computador mais parecia uma máquina de calcular. O aprendizado de novos conceitos, como a estrutura de seleção, nos dará uma visão maior da complexidade de tarefas que ele poderá executar.

Vamos refletir sobre a importância dessa estrutura, lendo com atenção as afirmativas a seguir:

1. Distribuição gratuita de cestas básicas.
2. Distribuição gratuita de cestas básicas para famílias com 4 ou mais componentes.
3. Distribuição gratuita de ingressos para o teatro, sendo dois para pessoas do sexo feminino e um para pessoas do sexo masculino.

Se observarmos essas afirmativas podemos concluir que:

- Na primeira, todas as pessoas recebem a cesta básica, o que equivaleria a um comando seqüencial.
- Na segunda, só recebem as cestas básicas as famílias com pelo menos quatro integrantes.
- Na terceira, dependendo do sexo, recebe-se um ou dois ingressos.

Assim, podemos avaliar a importância do teste nas duas últimas afirmativas, pois ações diferentes são executadas de acordo com o resultado.

Um exemplo do nosso dia-a-dia: imagine-se diante de um caixa eletrônico e suponha que sua senha seja 1234:

Na tela aparece a mensagem: O cursor ( █ ou   ) fica piscando:	Digite sua senha: █	
Você digita os algarismos da sua senha	1234	1233
Neste momento, a Unidade Aritmética e Lógica (um dos componentes da CPU) verifica se os números que você digitou são iguais a 1234. Caso tenham sido, aparece na tela a mensagem: VÁLIDA; mas se você digitou algo diferente, aparece na tela a mensagem: INVÁLIDA.	VÁLIDA	INVÁLIDA

**Conceito** é uma estrutura de controle de fluxo, executando um ou vários comandos se a condição testada for verdadeira e, em alguns casos, executando um ou vários comandos se for falsa.

## SINTAXES

### Seleção Simples

```
se ( condição )
{
    comando ;   ou
    < seqüência de comandos separados por ; >
}
```

A sintaxe acima representa a afirmativa 2, pois se a família tiver, no mínimo, quatro componentes, recebe a cesta básica; mas se a família tiver menos que quatro componentes, não recebe nada.

### Seleção Composta

```
se ( condição )
{
    comando ;   ou
    < seqüência de comandos separados por ; >
}
senao
{
    comando ;   ou
    < seqüência de comandos separados por ; >
}
```

A sintaxe acima representa a afirmativa 3 onde, dependendo do sexo, recebe-se um ou dois convites.

## OBSERVAÇÕES

1. Podemos constatar que esta estrutura faz parte do nosso cotidiano:
  - Se eu não tiver prova, vou à praia; senão vou estudar.
  - Se eu tiver aumento, troco de carro; senão espero o 13º salário.
  - Se minha média for maior ou igual a sete, passo direto; senão vou à prova final.
2. A única coisa diferente é a forma como iremos escrevê-la, pois as chaves { e } são obrigatórias uma vez que delimitam os comandos que pertencem a cada bloco, assim como os parênteses ( e ) que delimitam a condição.
3. Essa é uma estrutura muito importante em algoritmos porque nos dá a possibilidade de verificar o que foi digitado pelo usuário ou qual o conteúdo de uma variável após um processamento etc.

Vamos analisar cada linha da segunda sintaxe que é a mais completa, pois um ou vários comandos serão executados caso a condição testada seja verdadeira e outros comandos serão executados caso a condição seja falsa.

### 1ª linha: se (condição)

#### Condição

- A condição é uma expressão lógica testada pela Unidade Aritmética e Lógica, devolvendo como resposta: verdadeiro ou falso.

Convém aqui recordar os operadores relacionais e os operadores lógicos:

Operadores Relacionais	Usaremos	Operadores lógicos	Usaremos
igual	<code>==</code>	conjunção (e)	<code>&amp;&amp;</code>
diferente	<code>&lt; &gt;</code>	disjunção (ou)	<code>  </code>
maior	<code>&gt;</code>	negação (não)	<code>!</code>
menor que	<code>&lt;</code>		
maior ou igual a	<code>&gt;=</code>		
menor ou igual a	<code>&lt;=</code>		

a. A condição pode ser uma simples *expressão relacional* formada de *dois operandos do mesmo tipo* e de *um operador relacional* ( $>$ ,  $<$ ,  $\geq$ ,  $\leq$ ,  $\neq$  e  $\neq$ ).

A > B	lê-se: o conteúdo da variável A é maior do que o conteúdo da variável B?
A < 12	lê-se: o conteúdo da variável A é menor do que 12?
resp $\neq$ "S"	lê-se: o conteúdo da variável resp é diferente da letra S?
resp == "BRASIL"	lê-se: o conteúdo da variável resp é igual a BRASIL?

b. A condição pode ser uma *expressão lógica* formada de *pelo menos duas expressões relacionais* que precisarão ser unidas por um dos operadores lógicos ( $\&\&$  ou  $\|$ ).

A $\geq$ 1 $\&\&$ A $<$ 9	lê-se: o conteúdo da variável A é maior ou igual a 1 e menor que 9?
resp == "S" $\ $ resp == "s"	lê-se: o conteúdo da variável resp é igual a S ou igual a s?

Você deve estar perguntando: e os parênteses, não são necessários ?

Não, porque as operações aritméticas têm maior prioridade, depois as relacionais e por último as lógicas; mas, *cuidado*, nem todas as linguagens agem dessa maneira.

Você deverá ter uma atenção especial quando unir vários operadores lógicos para fazer seus testes. Veremos isso mais adiante.

#### 2<sup>a</sup> linha: {

Indica o início do bloco caso a condição testada seja verdadeira.

#### 3<sup>a</sup> linha: comando ; ou < seqüência de comandos separados por ; >

Nesta parte, são relacionados os comandos que serão executados caso a condição seja verdadeira.

#### 4<sup>a</sup> linha: }

Indica o fim do bloco caso a condição testada seja verdadeira.

#### 5<sup>a</sup> linha: senao

Este comando faz parte da estrutura do se e só deverá ser usado quando pelo menos uma ação tiver de ser executada se a condição testada for falsa.

## **6<sup>a</sup> linha: {**

Indica o início do bloco caso a condição testada seja falsa.

## **7<sup>a</sup> linha: comando ; ou < seqüência de comandos separados por ; >**

Nesta parte, são relacionados os comandos que serão executados caso a condição seja falsa.

## **8<sup>a</sup> linha: }**

Indica o fim do bloco caso a condição testada seja falsa.

### *A execução do comando se:*

1. Primeiro, a condição é avaliada pela Unidade Aritmética e Lógica, sendo a condição uma expressão que possa retornar um valor lógico (verdadeiro ou falso).

2. Se o valor retornado for verdadeiro, todos os comandos que se encontram entre o primeiro par de chaves serão executados; após a execução dos comandos, o fluxo do algoritmo passa para o primeiro comando depois do fechamento do bloco do **senao**, representado pelo símbolo } (se existir; caso contrário, será executado o primeiro comando depois de } do próprio bloco).

3. Se o valor retornado for falso, todos os comandos que se encontram entre o primeiro par de chaves serão ignorados e, se existir o comando **senao**, serão executados os comandos que se encontram entre o segundo par de chaves; caso contrário, nada acontecerá e o fluxo do algoritmo seguirá para o próximo comando.

Vejamos alguns exemplos:

## algoritmo 82

Ler um número e se ele for maior do que 20, então imprimir a metade do número.

Solução nº 1:

```
prog metade
    real numero, met;
    imprima "digite numero: ";
    leia numero;
    se( numero > 20. )
    {
        met <- numero / 2;
        imprima "\nmetade: ", met ;
    }
    imprima "\n";
fimprog
```

Nesta solução foi criada a variável met.

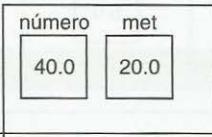
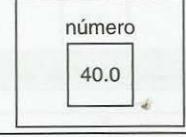
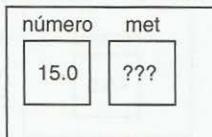
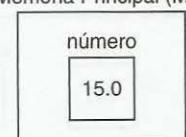
Solução nº 2:

```
prog metade
    real numero;
    imprima "digite numero: ";
    leia numero;
    se( numero > 20. )
    {
        imprima "\nmetade:",numero/2;
    }
    imprima "\n";
fimprog
```

Nesta solução não foi criada a variável met.

A linha de comando **imprima "\n"**; evita que o prompt fique na mesma linha do resultado.

A saída na tela será a mesma, mas, na MP, teremos duas variáveis. Para este tipo de exercício, a criação da variável met seria desnecessária.

VÍDEO	VÍDEO
Digite numero:40. metade:20.0	Digite numero:40. metade:20.0
MP Memória Principal (MP) 	MP Memória Principal (MP) 
VÍDEO	VÍDEO
Digite numero:15.	Digite numero:15.
MP Memória Principal (MP) 	MP Memória Principal (MP) 

## algoritmo 83

Só em novas

Ler um número e, se ele for positivo, imprimir seu inverso; caso contrário, imprimir o valor absoluto do número.

Solução nº 1:

```
prog inversoabsoluto
    real numero, inverso, absoluto;
    imprima "digite numero:";
    leia numero;
    se( numero > 0. )
        { inverso <- 1 / numero;
            imprima "\ninverso: ",
            inverso;}
    senao
        { absoluto <- numero * -1;
        # ou absoluto <- abs(numero);
            imprima "\nabsoluto: ",
            absoluto;}
    imprima "\n";
fimprog
```

Solução nº 2:

```
prog inversoabsoluto
    real numero;
    imprima "digite numero:";
    leia numero;
    se( numero > 0. )
        {imprima "\n\ninverso: ",1 /
        numero; }
    senao
        {imprima "absoluto: ", numero *
        (-1);
        #imprima "absoluto: ",
        abs(numero); }
    imprima "\n";
fimprog
```

VÍDEO	VÍDEO								
Digite numero:5. inverso:0.2	Digite numero:5. inverso:0.2								
MP Memória Principal (MP) <table border="1"><tr><td>número</td><td>inverso</td><td>absoluto</td></tr><tr><td>5.0</td><td>0.2</td><td>???</td></tr></table>	número	inverso	absoluto	5.0	0.2	???	MP Memória Principal (MP) <table border="1"><tr><td>número</td></tr><tr><td>5.0</td></tr></table>	número	5.0
número	inverso	absoluto							
5.0	0.2	???							
número									
5.0									
Digite numero:-5. absoluto:5.0	Digite numero:-5. absoluto:5.0								
MP Memória Principal (MP) <table border="1"><tr><td>número</td><td>inverso</td><td>absoluto</td></tr><tr><td>-5.0</td><td>???</td><td>5.0</td></tr></table>	número	inverso	absoluto	-5.0	???	5.0	MP Memória Principal (MP) <table border="1"><tr><td>número</td></tr><tr><td>-5.0</td></tr></table>	número	-5.0
número	inverso	absoluto							
-5.0	???	5.0							
número									
-5.0									

## algoritmo 84

Ler um número e imprimir se ele é par ou ímpar.

```
prog parimpar
    int a;
    imprima "\nDigite numero: ";
    leia a;
    se(a % 2 == 0)
        {imprima "\nPAR";}
    senao
        {imprima "\nIMPAR";}
    imprima "\n";
fimprog
```

### VÍDEO

Digite numero:24

PAR

Digite numero:25

IMPAR

Se você já entendeu tudo o que foi explicado até aqui, então será capaz de deduzir o que ficará armazenado nas variáveis do trecho do algoritmo a seguir, sabendo-se que elas tanto podem ser do tipo **int**, **real** ou **string**.

### DESAFIO

```
se(a > b)
{ aux <- a;
  a <- b;
  b <- aux;
}
se(a > c)
{ aux <- a;
  a <- c;
  c <- aux;
}
```

```
se(b > c)
{ aux <- b;
  b <- c;
  c <- aux;
}
.
.
.
```

Resposta: Na variável *a*, ficará o menor; na variável *b*, o do meio; e, na variável *c*, o maior (se forem números); ou então ficarão as três palavras em ordem alfabética, uma vez que também é uma ordem crescente.

## SES ANINHADOS (ENCAIXADOS)

Muitas vezes, em algumas aplicações, sentiremos a necessidade de tomar outras decisões dentro de uma das alternativas da estrutura do **se**; a isso chamamos de ses aninhados.

Vejamos um exemplo clássico de algoritmos:

### algoritmo 85

```
prog descubra
  real a, b, c, max;
  imprima "\ndigite 1 numero: ";
  leia a;
  imprima "\ndigite 2 numero: ";
  leia b;
  imprima "\ndigite 3 numero: ";
  leia c;
  se (a > b)
  {
    se (a > c)
    { max <- a ; }
    senao
    { max <- c; }
  }
  senao
  {
    se ( b > c)
    { max <- b; }
    senao
    { max <- c ; }
  }
  imprima "\n",max;
  imprima "\n";
```

## Você descobriu o que faz esse algoritmo?

Resposta: Armazena na variável **max** o maior número entre 3, imprimindo-o.

Desafio: Normalmente, o uso de **ses** aninhados melhora a performance do algoritmo. Será que nesse caso aconteceu isso? Tente usar o trecho do desafio anterior e melhore esta solução. A resposta estará mais adiante.

Outros exemplos:

### algoritmo 86

Ler um número e imprimir se ele é positivo, negativo ou nulo.

```
prog pnn
real num;
imprima "\nDigite numero: ";
leia num;
se(num > 0.)
{imprima "\nPOSITIVO";}
senao
{ se(num < 0.)
{ imprima "\nNEGATIVO";}
senao
{ imprima "\nNULO";}
}
imprima "\n";
fimprog
```

### VÍDEO

Digite numero:34.

POSITIVO

Digite numero:-12.

NEGATIVO

Digite numero:0.

NULO

Você deve estar fazendo algumas perguntas:

1. Por que não se perguntou se o número era igual a zero?

Resposta: Quando temos a possibilidade de três respostas, só precisamos fazer | 69

duas perguntas, pois a segunda pergunta nos dá as duas últimas respostas. Veja bem: se descartarmos a possibilidade de o número ser maior do que 0, ficamos com duas possibilidades: o número ser igual a 0 ou menor do que 0; dessa forma, uma pergunta é satisfatória.

### 2. Mas se eu fizer, estarei errado(a)?

Resposta: Não, mas não é necessário.

### 3. Por que a solução não poderia ser como a seguir?

```
prog se
    real num;
    imprima "\nDigite numero: ";
    leia num;
    se(num > 0.)
    { imprima "\nPOSITIVO";}
    se(num < 0.)
    { imprima "\nNEGATIVO";}
    se(num == 0.)
    { imprima "\nNULO";}
    imprima "\n";
fimprog
```

Resposta: Esta solução, embora você consiga atingir os objetivos do algoritmo, apresenta um grande inconveniente: sempre serão executados três testes, mesmo quando já tivermos classificado o número. Entretanto, na 1<sup>a</sup> solução, outro teste só será executado se ainda não tivermos chegado a uma conclusão sobre o número.

Esta estrutura precisa de vários ciclos para ser executada; portanto, evite usá-la desnecessariamente.

### algoritmo 87

Criar um algoritmo que permita ao aluno responder qual a capital do Brasil. Todas as possibilidades deverão ser pensadas.

```
prog geo
    string resp;
    imprima "\nQual a capital do Brasil? ";
    leia resp;
    se(resp == "BRASÍLIA" || resp == "Brasília")
    { imprima "\nPARABÉNS!"}
    senao
    { se(resp=="brasília" || resp=="BRASÍLIA"|| resp=="Brazília" || resp=="brazília")
      { imprima "\nCERTO! Mas atenção para grafia: Brasília ou BRASÍLIA "}
    senao
    { imprima "\nERRADO! ESTUDE MAIS!"}
```

```
}

imprima "\n";
fimprog
```

↳ Digite no ambiente Linux por causa da acentuação.

## VÍDEO

```
Qual a capital do Brasil? brasília
CERTO! Mas atenção para grafia: Brasília ou BRASÍLIA

Qual a capital do Brasil? BRASÍLIA
PARABÉNS!

Qual a capital do Brasil? Brasília
PARABÉNS!

Qual a capital do Brasil? Brazília
CERTO! Mas atenção para grafia: Brasília ou BRASÍLIA

Qual a capital do Brasil? BRAZÍLIA
CERTO! Mas atenção para grafia: Brasília ou BRASÍLIA

Qual a capital do Brasil? brazília
CERTO! Mas atenção para grafia: Brasília ou BRASÍLIA

Qual a capital do Brasil? Vitória
ERRADO! ESTUDE MAIS!
```

## algoritmo 88

### Algoritmo Calculadora

```
prog calculadora
    string resp;
    real a,b;
    imprima "\n\n\t\t\t*****\n";
    imprima "\t\t\tCALCULADORA*\n";
    imprima "\t\t\t*****\n";
    imprima "\n\t\t\t+ para somar\n";
    imprima "\n\t\t\t- para subtrair\n";
    imprima "\n\t\t\t* para multiplicar\n";
    imprima "\n\t\t\t/ para dividir\n";
    imprima "\n\n\t\t\tDigite opção: ";
    leia resp;
```

```

se(resp=="+")
{
    imprima "\nDigite 1 numero com ponto: ";
    leia a;
    imprima "\nDigite 2 numero com ponto: ";
    leia b;
    imprima "\nSOMA: ",a + b;
}
senao
{
    se(resp=="-")
    {
        imprima "\nDigite 1 numero com ponto: ";
        leia a;
        imprima "\nDigite 2 numero com ponto: ";
        leia b;
        imprima "\nSUBTRACAO: ",a - b;
    }
    senao
    {
        se(resp=="*")
        {
            imprima "\nDigite 1 numero com ponto: ";
            leia a;
            imprima "\nDigite 2 numero com ponto: ";
            leia b;
            imprima "\nMULTIPLICACAO: ",a * b;
        }
        senao
        {
            se(resp=="/")
            {
                imprima "\nDigite 1 numero com ponto: ";
                leia a;
                imprima "\nDigite 2 numero com ponto: ";
                leia b;
                imprima "\nDIVISAO: ",a / b;
            }
            senao
            {
                imprima "\nOPCAO NAO DISPONIVEL!";
            }
        }
    }
}
imprima "\n";
fimprog

```

## VÍDEO

\*\*\*\*\*  
\*CALCULADORA\*  
\*\*\*\*\*

- + para somar
- para subtrair
- \* para multiplicar
- / para dividir

Digite opção:\*

Digite 1 numero com ponto:23.

Digite 2 numero com ponto:12.

\*\*\*\*\*  
\*CALCULADORA\*  
\*\*\*\*\*

- + para somar
- para subtrair
- \* para multiplicar
- / para dividir

Digite opção:&

OPCAO NAO DISPONIVEL!

## UM POUCO MAIS SOBRE VARIÁVEL STRING

Já vimos que a variável simples string é armazenada na memória como se fosse um vetor onde cada caractere é armazenado em uma posição conforme o esquema a seguir:

*variável string*

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7	byte 8	byte 9	.....	byte n	

Analisando sob esse prisma, podemos dizer que a variável simples string é um conjunto de n variáveis de um único caractere e todas com o mesmo nome.

Quando se compararam duas variáveis strings, na verdade, estamos comparando os códigos dos caracteres de cada string, baseado em seus va-

lores no código ASCII (Apêndice II). Dessa forma, não é difícil entender o uso dos sinais de **>**, **<** ou de **=**.

Usar o sinal de **>** significa perguntar se *vem depois na ordem alfabética* (ou **<** vem antes), uma vez que o código ASCII está em ordem alfabética, isto é, a letra A tem um código menor do que a letra B e assim sucessivamente. Veja o trecho da tabela a seguir:

CARACTERE	Código ASCII em decimal
A	65
B	66
C	67
...	...
L	76
...	...
Z	90
...	...
a	97
b	98
c	99
...	...
l	208
...	...
z	122

Observando o trecho da tabela acima, você poderá compreender por que uma palavra escrita em letras maiúsculas (ALGORITMOS) não é considerada igual a uma palavra escrita em letras minúsculas (algoritmos):

NOME1												
65	76	71	79	82	73	84	77	79	83	\0	?	
0	1	2	3	4	5	6	7	8	9	!	0	

NOME2												
97	108	103	111	114	105	116	109	111	115	\0	?	
0	1	2	3	4	5	6	7	8	9	!	0	

## ALGUMAS QUESTÕES

1. Algumas linguagens permitem que se monte uma expressão relacional com os nomes das variáveis:

COMPARAÇÃO
se (NOME1 > NOME2)

Outras linguagens não permitem o uso dos operadores  $>$ ,  $<$  ou  $=$  e oferecem funções que viabilizam esta operação:

se (strcmp( NOME1, NOME2) == "igual" )	Essa expressão verifica se os conteúdos das variáveis são iguais.
se (strcmp ( NOME1, NOME2) == "maior" )	Essa expressão verifica se o conteúdo da 1 <sup>a</sup> variável, na ordem alfabética, vem depois do conteúdo da 2 <sup>a</sup> variável.
se (strcmp ( NOME1, NOME2) == "menor")	Essa expressão verifica se o conteúdo da 1 <sup>a</sup> variável, na ordem alfabética, vem antes do conteúdo da 2 <sup>a</sup> variável.

---

↳ No interpretador que sugerimos, você poderá fazer das duas maneiras.

---

2. Mas, independente do exposto acima, como a Unidade Aritmética e Lógica compara esses dados? Vejamos o exemplo a seguir, sabendo-se que a variável NOME1 armazena a palavra *caso* e a variável NOME2 armazena a palavra *casa*:

NOME1						
99	97	115	111	\0	?	
0	1	2	3	4	5	
NOME2						
99	97	115	97	\0	?	
0	1	2	3	4	5	

Se usássemos a expressão Se (NOME1 < NOME2), a Unidade Aritmética e Lógica faria subtrações, considerando como minuendo o 1<sup>º</sup> código de NOME1 e, como subtraendo, o 1<sup>º</sup> código de NOME2 e, assim sucessivamente, até que o resultado fosse diferente de 0.

*1º passo:*

$$99 - 99 = 0$$

continuo porque o resultado foi 0 (“igual”).

*2º passo:*

$$97 - 97 = 0$$

continuo porque o resultado foi 0 (“igual”).

*3º passo:*

$$115 - 115 = 0$$

continuo porque o resultado foi 0 (“igual”).

*4º passo:*

$$111 - 97 = 14$$

paro e vou verificar o que foi pedido

Como foi perguntado se NOME1 > NOME2 e tenho 14 (“maior”) como resposta, vou dizer que é VERDADE. (*Esse seria o “raciocínio” da UAL.*)

**3.** Algumas linguagens permitem que se atribua uma variável caracter a outra variável caracter enquanto outras oferecem uma função para produzir o mesmo efeito:

ATRIBUIÇÃO	FUNÇÃO
AUX <- NOME1;	copia(AUX, NOME1);

**4.** Algumas linguagens têm o operador // ou + ( coloca o conteúdo de uma após o da outra):

CONCATENAÇÃO e ATRIBUIÇÃO*	
Em algumas linguagens	Em nosso estudo
NOME <- NOME1 // NOME2;	NOME <- strconcat(NOME1,NOME2);

Outras linguagens não permitem nenhuma das formas acima e oferecem funções que viabilizam estas operações:

CONCATENAÇÃO e ATRIBUIÇÃO
copia(NOME, NOME1);
concatena(NOME, NOME2);

*1º passo:*

$$99 - 99 = 0$$

continuo porque o resultado foi 0 (“igual”).

*2º passo:*

$$97 - 97 = 0$$

continuo porque o resultado foi 0 (“igual”).

*3º passo:*

$$115 - 115 = 0$$

continuo porque o resultado foi 0 (“igual”).

*4º passo:*

$$111 - 97 = 14$$

paro e vou verificar o que foi pedido

Como foi perguntado se NOME1 > NOME2 e tenho 14 (“maior”) como resposta, vou dizer que é VERDADE. (*Esse seria o “raciocínio” da UAL.*)

**3.** Algumas linguagens permitem que se atribua uma variável caracter a outra variável caracter enquanto outras oferecem uma função para produzir o mesmo efeito:

ATRIBUIÇÃO	FUNÇÃO
AUX <- NOME1;	copia(AUX, NOME1);

**4.** Algumas linguagens têm o operador // ou + ( coloca o conteúdo de uma após o da outra):

CONCATENAÇÃO e ATRIBUIÇÃO*	
Em algumas linguagens	Em nosso estudo
NOME <- NOME1 // NOME2;	NOME <- strconcat(NOME1,NOME2);

Outras linguagens não permitem nenhuma das formas acima e oferecem funções que viabilizam estas operações:

CONCATENAÇÃO e ATRIBUIÇÃO
copia(NOME, NOME1);
concatena(NOME, NOME2);

## ALTERNATIVA DE MÚLTIPLAS ESCOLHAS

É uma alternativa para os *ses aninhados*, deixando o algoritmo com uma estrutura melhor.

Sintaxe:

```
escolha (expressão)
{
    caso <rótulo 1> : comando1;
                        comando2;
                        pare;
    caso <rótulo 2> : comando1;
                        comando2;
                        pare;
    caso <rótulo n> : comando1;
                        comando2;
                        pare;
    senao comando;
}
```

Considerações:

1. A expressão é avaliada e o valor será comparado com um dos rótulos.
2. A opção *senao* é opcional.
3. O rótulo será aqui definido como uma *constante caracter* (*de um caracter*) ou uma *constante numérica inteira*, embora em algumas linguagens possam ser usadas constantes caracter com mais de um caracter.
4. A estrutura é muito usada em algoritmos com menus, tornando-os mais claros do que quando usamos *ses aninhados*.
5. O interpretador que sugerimos nessa versão não apresenta essa estrutura.

Exemplo:

Escrever um algoritmo que leia um peso na Terra e o número de um planeta e imprima o valor do seu peso neste planeta. A relação de planetas é dada a seguir juntamente com o valor das gravidades relativas à Terra:

#	<i>gravidade relativa</i>	<i>Planeta</i>
1	0,37	Mercúrio
2	0,88	Vênus
3	0,38	Marte
4	2,64	Júpiter
5	1,15	Saturno
6	1,17	Urano

Para calcular o peso no planeta use a fórmula:

$$P_{\text{planeta}} = \frac{P_{\text{terra}}}{10} \cdot \text{gravidade}$$

### algoritmo 89

```
prog pesodoplaneta
    int op;
    real pterra;
    imprima " planetas que podem ser analisados: ";
    imprima " 1 mercurio ";
    imprima " 2 venus ";
    imprima " 3 marte ";
    imprima " 4 jupiter ";
    imprima " 5 saturno ";
    imprima " 6 urano ";
    imprima " escolha o planeta a ser analisado ";
    leia op ;
    imprima " entre com um peso na terra ";
    leia pterra;
    escolha ( op)
    {
        caso 1: imprima "seu peso no planeta terra é: ", (pterra/10)*0.37 ;pare;
        caso 2: imprima "seu peso no planeta terra é: ", (pterra/10)*0.88;pare;
        caso 3: imprima "seu peso no planeta terra é: ", (pterra/10)*0.38;pare;
        caso 4: imprima "seu peso no planeta terra é: ", (pterra/10)*2.64;pare;
        caso 5: imprima "seu peso no planeta terra é: ", (pterra/10)*1.15;pare;
        caso 6: imprima "seu peso no planeta terra é: ", (pterra/10)*1.17;pare;
        senao: imprima "este planeta não pode ser analisado";
    }
    imprima "\n";
fimprog
```

## EXERCÍCIOS – LISTA 2

### ESTRUTURA DO SE

### algoritmo 90

Entrar com um número e imprimi-lo caso seja maior que 20.

```
prog sel
    real numero;
    imprima "\n\ndigite numero: ";
    leia numero;
    se ( numero > 20. )
    {
        imprima numero;
        imprima "\n";
    }
fimprog
```

## algoritmo 91

Construir um algoritmo que leia dois valores numéricos inteiros e efetue a adição; caso o resultado seja maior que 10, apresentá-lo.

```
prog se2
    int num1, num2, soma;
    imprima "\n\nDigite 1 numero: ";
    leia num1;
    imprima "\n\nDigite 2 numero: ";
    leia num2;
    soma <- num1 + num2;
    se ( soma > 10 )
    {   imprima "\nsoma: ", soma; }
    imprima "\n";
fimprog
```

## algoritmo 92

Construir um algoritmo que leia dois números e efetue a adição. Caso o valor somado seja maior que 20, este deverá ser apresentado somando-se a ele mais 8; caso o valor somado seja menor ou igual a 20, este deverá ser apresentado subtraindo-se 5.

```
prog se3
    real num1, num2, soma;
    imprima "\ndigit 1 numero: ";
    leia num1;
    imprima "\ndigit 2 numero: ";
    leia num2;
    soma <- num1 + num2;
    se ( soma > 20. )
    {   imprima "\nsoma: ", soma + 8; }
    senao
    {   imprima "\nsoma: ", soma - 5; }
    imprima "\n";
fimprog
```

## algoritmo 93

Entrar com um número e imprimir a raiz quadrada do número caso ele seja positivo e o quadrado do número caso ele seja negativo.

```
prog se4
    real numero;
    imprima "\ndigit numero: ";
    leia numero;
    se ( numero > 0. )
    {   imprima "\nraiz: ", raiz(numero); }
```

```
senao
{
    se ( numero < 0. )
    {   imprima "\nquadrado: ", numero ** 2; }
}
imprima "\n";
fimprog
```

### algoritmo 94

---

*Entrar com um número e imprimir uma das mensagens: é multiplo de 3 ou não é multiplo de 3.*

```
prog se5
int numero;
imprima "\ndigite numero: ";
leia numero;
se ( numero % 3 == 0 )
{   imprima "\nmultiplo de 3";}
senao
{   imprima "\nnão é multiplo de 3"; }
imprima "\n";
fimprog
```

### algoritmo 95

---

*Entrar com um número e informar se ele é ou não divisível por 5.*

```
prog se6
int numero;
imprima "\ndigite numero: ";
leia numero;
se ( numero % 5 == 0 )
{   imprima "\ne divisivel por 5";}
senao
{   imprima "\nnão é divisivel por 5"; }
imprima "\n";
fimprog
```

### algoritmo 96

---

*Entrar com um número e informar se ele é divisível por 3 e por 7.*

```
prog se7
int numero;
imprima "\ndigite numero: ";
leia numero;
se (numero %3==0 && numero %7==0 ) /*poderia testar (numero %21 == 0) */
{   imprima "\ndivisível por 3 e por 7"; }
senao
```

```
{ imprima "\nnão é divisível por 3 e por 7"; }  
imprima "\n";  
fimprog
```

## algoritmo 97

---

Entrar com um número e informar se ele é divisível por 10, por 5, por 2 ou se não é divisível por nenhum destes.

```
prog se8  
int numero;  
imprima "\ndigite numero: ";  
leia numero;  
se ( numero % 10 == 0 )  
{ imprima "\nmúltiplo de 10"; }  
senao  
{  
    se ( numero % 2 == 0 )  
    { imprima "\nmúltiplo de 2"; }  
    senao  
    { imprima "\nnão é múltiplo de 2 nem de 5"; }  
}  
}  
imprima "\n";  
fimprog
```

## algoritmo 98

---

A prefeitura do Rio de Janeiro abriu uma linha de crédito para os funcionários estatutários. O valor máximo da prestação não poderá ultrapassar 30% do salário bruto. Fazer um algoritmo que permita entrar com o salário bruto e o valor da prestação e informar se o empréstimo pode ou não ser concedido.

```
prog se9  
real sb, vp;  
imprima "\ndigite o salario: ";  
leia sb;  
imprima "\ndigite o valor da prestacao: ";  
leia vp;  
se( vp <= 0.3 * sb )  
{ imprima "\nemprestimo concedido ";}  
senao  
{ imprima "\nemprestimo negado ";}  
imprima "\n";  
fimprog
```

## algoritmo 99

Ler um número inteiro de 3 casas decimais e imprimir se o algarismo da casa das centenas é par ou ímpar.

```
prog sel0
    int num, c;
    imprima "\nnumero de 3 algarismos: ";
    leia num;
    c <- num div 100;
    se( c % 2 == 0 )
    {imprima "\no algarismo das centenas e par: ",c;};
    senao
    {imprima "\no algarismo das centenas e impar: ",c;};
    imprima "\n";
fimprog
```

## algoritmo 100

Ler um número inteiro de 4 casas e imprimir se é ou não múltiplo de quatro o número formado pelos algarismos que estão nas casas das unidades de milhar e das centenas.

```
prog sel1
    int num, c;
    imprima "\nnumero de 4 algarismos: ";
    leia num;
    c <- num div 100;
    se( c % 4 == 0 )
    {imprima "\no numero e multiplo de 4: ",c;};
    senao
    {imprima "\no numero nao e multiplo de 4: ",c;};
    imprima "\n";
fimprog
```

## algoritmo 101

Construir um algoritmo que indique se o número digitado está compreendido entre 20 e 90 ou não.

```
prog sel2
    real num;
    imprima "\ndigite numero: ";
    leia num;
    se ( num > 20. && num < 90. )
    { imprima "\no numero esta na faixa de 20 a 90, exclusive"; }
    senao
    { imprima "\no numero esta fora da faixa de 20 a 90"; }
    imprima "\n";
fimprog
```

## algoritmo 102

---

*Entrar com um número e imprimir uma das mensagens: maior do que 20, igual a 20 ou menor do que 20.*

```
prog se13
    real numero;
    imprima "\ndigite numero: ";
    leia numero;
    se ( numero > 20. )
    { imprima "\nmaior que 20"; }
    senao
    {
        se ( numero < 20. )
        { imprima "\nmenor que 20"; }
        senao
        { imprima "\nigual a 20"; }
    }
    imprima "\n";
fimprog
```

## algoritmo 103

---

*Entrar com o ano de nascimento de uma pessoa e o ano atual. Imprimir a idade da pessoa. Não se esqueça de verificar se o ano de nascimento é um ano válido.*

```
prog se14
    int anon, anoa;
    imprima "\nEnter com ano atual: ";
    leia anoa;
    imprima "\nEnter com ano de nascimento: ";
    leia anon;
    se ( anon > anoa )
    { imprima "\nAno de Nascimento Invalido"; }
    senao
    { imprima "\nIdade: " , anoa - anon; }
    imprima "\n";
fimprog
```

## algoritmo 104

---

*Entrar com nome, sexo e idade de uma pessoa. Se a pessoa for do sexo feminino e tiver menos que 25 anos, imprimir nome e a mensagem: ACEITA. Caso contrário, imprimir nome e a mensagem: NÃO ACEITA. (Considerar f ou F.)*

```
prog se15
    int idade;
    string nome, sexo;
    imprima "\ndigite nome: ";
```

```
leia nome;
imprima "\ndigite sexo: ";
leia sexo;
imprima "\ndigite idade: ";
leia idade;
se( (sexo == "F" || sexo=="f") && idade<25)
{ imprima "\n", nome, " ACEITA"; }
senao
{ imprima "\n", nome, " NAO ACEITA"; }
imprima "\n";
fimprog
```

## algoritmo 105

---

*Entrar com a sigla do estado de uma pessoa e imprimir uma das mensagens:*

- carioca
- paulista
- mineiro
- outros estados

```
prog sel6
string sigla;
imprima "\ndigite sigla: ";
leia sigla;
se ( sigla == "RJ" || sigla == "rj")
{ imprima "\ncarioca"; }
senao
{ se (sigla == "SP" || sigla == "sp")
{ imprima "\npaulista"; }
senao
{ se ( sigla == "MG" || sigla == "mg")
{ imprima "\nmineiro"; }
senao
{ imprima "\noutros estados"; }
}
imprima "\n";
fimprog
```

## algoritmo 106

---

*Entrar com um nome e imprimi-lo se o primeiro caractere for a letra A (considerar letra minúscula ou maiúscula).*

```
prog sel7
string nome, letra;
imprima "\ndigite nome: ";
```

```
leia nome;
letra <- strprim(nome);
se ( letra == "A" || letra == "a" )
{ imprima "\n", nome; }
imprima "\n";
fimprog
```

## algoritmo 107

---

*Entrar com o nome de uma pessoa e só imprimi-lo se o prenome for JOSÉ.*

```
prog se18
string nome;
imprima "\ndigite nome: ";
leia nome;
se (strnprim(nome, 5) == "JOSÉb" || nome=="JOSÉ" )
{ imprima "\n", nome; }
imprima "\n";
fimprog
```

---

⇨ b significa pressionar a barra de espaço

---

⇨ Há duas possibilidades, pois pode ser que a pessoa só digite JOSÉ e não o nome completo.

---

## algoritmo 108

---

*Idem ao anterior, porém considerar: JOSÉ, José ou josé.*

```
prog se19
string nome, prenome;
imprima "\ndigite nome: ";
leia nome;
prenome <- strnprim(nome, 5);
se( prenome == "JOSÉb" || prenome == "Joséb" || prenome == "joséb"
|| nome=="josé" || nome == "José" || nome=="JOSÉ" ) #mesma linha
{ imprima "\n", nome; }
imprima "\n";
fimprog
```

## algoritmo 109

Criar um algoritmo que entre com dois nomes e imprimi-los em ordem alfabética.

```
prog se20a
    string nome1,nome2;
    imprima "\ndigite 1º nome: ";
    leia nome1;
    imprima "\ndigite 2º nome: ";
    leia nome2;
    se (nome1< nome2)
    { imprima "\n", nome1," ", 
    nome2 ; }
    senao
    { imprima "\n", nome2," ", 
    nome1; }
    imprima "\n";
fimprog
```

```
prog se20b
    string nome1,nome2;
    imprima "\ndigite 1º nome: ";
    leia nome1;
    imprima "\ndigite 2º nome: ";
    leia nome2;
    se (strcmp(nome1, nome2) ==
"menor")
    { imprima "\n", nome1," ", 
    nome2; }
    senao
    { imprima "\n", nome2," ", 
    nome1; }
    imprima "\n";
fimprog
```

## algoritmo 110

Criar um algoritmo que leia dois números e imprimir uma mensagem dizendo se são iguais ou diferentes.

```
prog se21
    real a, b;
    imprima "\ndigite 1º numero: ";
    leia a;
    imprima "\ndigite 2º numero: ";
    leia b;
    se (a == b)
    { imprima "\niguais"; }
    senao
    { imprima "\ndiferentes"; }
    imprima "\n";
fimprog
```

## algoritmo 111

Entrar com dois números e imprimir o maior número (suponha números diferentes).

```
prog se22
    real a, b;
    imprima "\ndigite 1º numero: ";
    leia a;
    imprima "\ndigite 2º numero: ";
```

```
leia b;
se (a > b)
{ imprima "\nmaior: ", a; }
senao
{ imprima "\nmaior: ", b; }
imprima "\n";
fimprog
```

### algoritmo 112

---

*Entrar com dois números e imprimir o menor número (suponha números diferentes).*

```
prog se23
real a, b;
imprima "\ndigite 1º numero: ";
leia a;
imprima "\ndigite 2º numero: ";
leia b;
se (a < b)
{ imprima "\nmenor: ", a; }
senao
{ imprima "\nmenor: ", b; }
imprima "\n";
fimprog
```

### algoritmo 113

---

*Entrar com dois números e imprimi-los em ordem crescente (suponha números diferentes).*

```
prog se24
real a, b;
imprima "\ndigite 1º numero: ";
leia a;
imprima "\ndigite 2º numero: ";
leia b;
se (a < b)
{ imprima "\n",a , " ",b; }
senao
{ imprima "\n",b , " ",a; }
imprima "\n";
fimprog
```

### algoritmo 114

---

*Entrar com dois números e imprimi-los em ordem decrescente (suponha números diferentes).*

```
prog se25
real a, b;
```

```

imprima "\ndigite 1º numero: ";
leia a;
imprima "\ndigite 2º numero: ";
leia b;
se (a > b)
{ imprima "\n",a , " ",b; }
senao
{ imprima "\n",b , " ",a; }
imprima "\n";
fimprog

```

### algoritmo 115

---

*Criar o algoritmo que deixe entrar com dois números e imprimir o quadrado do menor número e a raiz quadrada do maior número, se for possível.*

```

prog se26
real x, y;
imprima "\ndigite valor de x: ";
leia x;
imprima "\ndigite valor de y: ";
leia y;
se(x > y)
{imprima "\n", raiz(x), "\n", y **2;}
senao
{ se(y > x)
{imprima "\n", raiz(y), "\n", x **2;}
senao
{ imprima "\nnumeros iguais";}
}
imprima "\n";
fimprog

```

### algoritmo 116

---

*Entrar com três números e imprimir o maior número (suponha números diferentes).*

```

prog se27solucao1
real a, b, c;
imprima "\ndigite 1 numero: ";
leia a;
imprima "\ndigite 2 numero: ";
leia b;
imprima "\ndigite 3 numero: ";
leia c;
se (a > b)
{
  se (a > c)
  { imprima "\nmaior: ", a ; }
}

```

```

senao
{ imprima "\nmaior: ", c; }
}
senao
{
  se ( b > c )
  { imprima "\nmaior: ", b; }
  senao
  { imprima "\nmaior: ", c ; }
}
imprima "\n";
fimprog

prog se27solucao2
real a, b, c;
imprima "\ndigite 1 numero: ";
leia a;
imprima "\ndigite 2 numero: ";
leia b;
imprima "\ndigite 3 numero: ";
leia c;
se ( a > b && a > c )
{ imprima "\nmaior: ", a ; }
senao
{
  se ( b > c )
  { imprima "\nmaior: ", b ; }
  senao
  { imprima "\nmaior: ", c ; }
}
imprima "\n";
fimprog

```

### algoritmo 117

---

*Entrar com três números e armazenar o maior número na variável de nome maior (suponha números diferentes).*

---

↳ Esta é uma solução alternativa e melhor para o algoritmo anterior.

---

```

prog se28solucao1
real a, b, c, maior;
imprima "\ndigite 1 numero: ";
leia a;
imprima "\ndigite 2 numero: ";
leia b;
imprima "\ndigite 3 numero: ";
leia c;

```

```

se (a > b)
{ maior <- a; }
senao
{ maior <- b ; }
se (c > maior)
{ maior <- c; }
imprima "\nmaior: ", maior;
imprima "\n";
fimprog

prog se28solucao2
real a, b, c, maior;
imprima "\ndigite 1 numero: ";
leia a;
imprima "\ndigite 2 numero: ";
leia b;
imprima "\ndigite 3 numero: ";
leia c;
maior <- a;
se (b > maior)
{ maior <- b ; }
se (c > maior)
{ maior <- c; }
imprima "\nmaior: ", maior;
imprima "\n";
fimprog

```

### algoritmo 118

---

*Entrar com três números e imprimi-los em ordem crescente (suponha números diferentes).*

↳ Várias são as soluções para colocar três números em ordem crescente, decrescente, maior e menor ou intermediário. Acreditamos que esta solução seja bem simples e tem como objetivo armazenar na variável *a* o menor valor; na variável *b*, o segundo valor; e na variável *c*, o maior valor.

---

```

prog se29
real a, b, c, aux;
imprima "\ndigite 1 numero: ";
leia a;
imprima "\ndigite 2 numero: ";
leia b;
imprima "\ndigite 3 numero: ";
leia c;
se ( a > b )
{ aux <- a; a <- b; b <- aux; }

```

```

se ( a > c )
{ aux <- a; a <- c; c <- aux; }
se ( b > c )
{ aux <- b; b <- c; c <- aux; }
imprima "\nOrdem Crescente: ", a, " ", b, " ", c ;
imprima "\n";
fimprog

```

## algoritmo 119

*Entrar com três números e imprimi-los em ordem decrescente (suponha números diferentes).*

```

prog se30
real a, b, c, aux;
imprima "\ndigite 1 numero: ";
leia a;
imprima "\ndigite 2 numero: ";
leia b;
imprima "\ndigite 3 numero: ";
leia c;
se ( a > b )
{ aux <- a; a <- b; b <- aux; }
se ( a > c )
{ aux <- a; a <- c; c <- aux; }
se ( b > c )
{ aux <- b; b <- c; c <- aux; }
imprima "\nOrdem Decrescente: ", c, " ", b, " ", a ;
imprima "\n";
fimprog

```

### algoritmo 120

*Entrar com três números e armazená-los em três variáveis com os seguintes nomes: maior, intermediário e menor (suponha números diferentes).*

```
prog se31a
real a, b, c, maior, intermediario, menor;
imprima "\ndigite 1 numero: ";
leia a;
imprima "\ndigite 2 numero: ";
leia b;
imprima "\ndigite 3 numero: ";
leia c;
se ( a > b )
{ se ( c > a )
{ maior <- c;
intermediario <- a ;
menor <- b;
```

```

    }
senao
{ se ( c > b)
{ maior <- a;
intermediario <- c;
menor <- b;
}
senao
{ maior <- a;
intermediario <- b;
menor <- c ;
}
}
senao
{ se ( c >b)
{ maior <- c;
intermediario <- b;
menor <- a;
}
senao
{ se ( c > a)
{ maior <- b;
intermediario <- c;
menor <- a;
}
senao
{ maior <- b;
intermediario <- a;
menor <-c;
}
}
imprima "\nmaior: ", maior;
imprima "\nintermediario: ",intermediario;
imprima "\nmenor: ", menor;
imprima "\n";
fimprog

prog se31b
real a, b, c, aux, maior, intermediario,menor;
imprima "\ndigite 1 numero: ";
leia a;
imprima "\ndigite 2 numero: ";
leia b;
imprima "\ndigite 3 numero: ";
leia c;
se ( a > b )
{ aux <- a;      a <- b;      b <- aux; }

```

```

se ( a > c )
{   aux <- a;      a <- c;      c <- aux; }
se ( b > c )
{   aux <- b;      b <- c;      c <- aux; }
maior <- c;
intermediario <- b;
menor <- a;
imprima "\nmaior: ", maior;
imprima "\nintermediario: ", intermediario;
imprima "\nmenor: ", menor;
imprima "\n";
fimprog

```

## algoritmo 121

---

*Efetuar a leitura de cinco números inteiros diferentes e identificar o maior e o menor valor.*

```

prog se32
int n1, n2, n3, n4, n5, maior, menor;
imprima "\ndigite 1 numero: ";
leia n1;
imprima "\ndigite 2 numero: ";
leia n2;
imprima "\ndigite 3 numero: ";
leia n3;
imprima "\ndigite 4 numero: ";
leia n4;
imprima "\ndigite 5 numero: ";
leia n5;
se (n1 < > n2 && n1 < > n3 && n1 < > n4 && n1 < > n5 && n2 < > n3 && n2 < > n4
&& n2 < > n5 && n3 < > n4 && n3 < > n5 && n4 < > n5)
{
/* Teste necessário porque o exercício fala sobre os números serem
diferentes. Veja como é importante saber COMBINAÇÃO */
se (n1 > n2)
{   maior <- n1;    menor <- n2; }
senao
{   maior <- n2;    menor <- n1; }
se (n3 > maior)
{   maior <- n3; }
senao
{
    se (n3 < menor)
    {   menor <- n3; }
}
se (n4 > maior)
{   maior <- n4; }

```

```

senao
{
    se (n4 < menor)
    { menor <- n4; }
}
se (n5 > maior)
{ maior <- n5; }
senao
{
    se (n5 < menor)
    { menor <- n5; }
}
imprima "\nMaior= ", maior;
imprima "\nMenor= ", menor;
}
senao
{ imprima "\nOs valores devem ser diferentes ";}
imprima "\n";
fimprog

```

---

↳ Um algoritmo para achar o menor ou o maior com vários números fica extremamente longo. Quando você aprender as estruturas de repetição, verá que esse algoritmo poderá ser melhorado.

## algoritmo 122

Ler três números e imprimir se eles podem ou não ser lados de um triângulo.

```

prog se33
real a, b, c;
imprima "\ndigite 1 lado: ";
leia a;
imprima "\ndigite 2 lado: ";
leia b;
imprima "\ndigite 3 lado: ";
leia c;
se ( a < b + c && b < a + c && c < a + b )
{ imprima "\nPodem ser lados de um triangulo"; }
senao
{ imprima "\nNão podem ser lados de um triangulo"; }
imprima "\n";
fimprog

```

## algoritmo 123

Ler três números, os possíveis lados de um triângulo, e imprimir a classificação segundo os lados.

```

prog se34
real a, b, c;
imprima "\ndigite 1 lado: ";
leia a;
imprima "\ndigite 2 lado: ";
leia b;
imprima "\ndigite 3 lado: ";
leia c;
se ( a < b + c && b < a + c && c < a + b )
{
    se ( a == b && a == c )
    { imprima "\nTriangulo equilatero"; }
    senao
    {
        se ( a == b || a == c || b == c )
        { imprima "\nTriangulo isosceles"; }
        senao
        { imprima "\nTriangulo escaleno"; }
    }
}
senao
{ imprima "\nAs medidas não formam um triangulo"; }
imprima "\n";
fimprog

```

## algoritmo 124

---

Ler três números, os possíveis lados de um triângulo, e imprimir a classificação segundo os ângulos.

```

prog se35
real a, b, c, maior, lados;
imprima "\ndigite 1 lado: ";
leia a;
imprima "\ndigite 2 lado: ";
leia b;
imprima "\ndigite 3 lado: ";
leia c;
se ( a < b + c && b < a + c && c < a + b )
{
    se ( a > b && a > c )
    { maior <- a ** 2; lados <- b ** 2 + c ** 2; }
    senao
    { se ( b > c )
        { maior <- b ** 2; lados <- a ** 2 + c ** 2; }
        senao
        { maior <- c ** 2; lados <- a ** 2 + b ** 2; }
    }
}

```

```

se ( maior == lados )
{ imprima "\nTriangulo Retangulo"; }
senao
{ se ( maior > lados )
{ imprima "\nTriangulo Obtusangulo"; }
senao
{ imprima "\nTriangulo Acutangulo"; }
}
}
senao
{ imprima "\nas medidas não formam um triangulo"; }
imprima "\n";
fimprog

```

## algoritmo 125

---

*Entrar com a idade de uma pessoa e informar:*

- *se é maior de idade*
- *se é menor de idade*
- *se é maior de 65 anos*

```

prog se36
int idade;
imprima "\ndigite sua idade: ";
leia idade;
se( idade >= 65 )
{ imprima "\nmaior de 65"; }
senao
{ se ( idade >= 18 )
{ imprima "\nmaior de idade"; }
senao
{ imprima "\nmenor de idade"; }
}
imprima "\n";
fimprog

```

## algoritmo 126

---

*Ler um número e imprimir se ele é igual a 5, a 200, a 400, se está no intervalo entre 500 e 1000, inclusive, ou se ele está fora dos escopos anteriores.*

```

prog se37
real x;
imprima "\ndigite valor de X: ";
leia x ;
se ( x == 5. )
{ imprima "\n0 numero é o 5"; }

```

```

senao
{ se ( x ==200. )
{ imprima "\nO numero é o 200"; }
senao
{ se ( x == 400. )
{ imprima "O numero é o 400"; }
senao
{ se ( x >= 500. && x <= 1000.)
{ imprima "\nIntervalo 500-1000"; }
senao
{ imprima "\nFora do escopo"; }
}
}
imprima "\n";
fimprog

```

### algoritmo 127

---

*Entrar com nome, nota da PR1 e nota da PR2 de um aluno. Imprimir nome, nota da PR1, nota da PR2, média e uma das mensagens: Aprovado, Reprovado ou em Prova Final (a média é 7 para aprovação, menor que 3 para reprovação e as demais em prova final).*

```

prog se38
real nota1, nota2, media;
string nome;
imprima "\ndigite nome: ";
leia nome;
imprima "\ndigite 1ª nota: ";
leia nota1;
imprima "\ndigite 2ª nota: ";
leia nota2;
media <- (nota1 + nota2)/ 2;
se ( media >=7. )
{ imprima nome," ", media, " AP"; }
senao
{ se ( media < 3. )
{ imprima nome," ", media, " RP"; }
senao
{ imprima nome," ", media, " PF"; }
}
imprima "\n";
fimprog

```

## algoritmo 128

Entrar com um verbo no infinitivo e imprimir uma das mensagens:

- verbo não está no infinitivo
- verbo da 1<sup>a</sup> conjugação
- verbo da 2<sup>a</sup> conjugação
- verbo da 3<sup>a</sup> conjugação

```
prog se39
    string verbo, letra;
    int n;
    imprima "\ndigite verbo: ";
    leia verbo;
    letra <- strlft(verbo);
    se ( letra == "R" || letra == "r")
    {
        n <- strtam(verbo) - 2;
        letra <- strelem(verbo,n);
        se ( letra == "A" || letra == "a" )
        { imprima "\n1a conjugacao"; }
        senao
        {
            se ( letra == "E" || letra == "e" || letra == "o" || letra == "O" )
            { imprima "\n2a conjugacao"; }
            senao
            {
                se ( letra == "I" || letra == "i" )
                { imprima "\n3a conjugacao"; }
                senao
                { imprima "\nnao existe verbo com terminacao ur"; }
            }
        }
    }
    senao
    { imprima "\nnao e um verbo no infinitivo"; }
    imprima "\n";
fimprog
```

## algoritmo 129

Entrar com o salário de uma pessoa e imprimir o desconto do INSS segundo a tabela a seguir:

menor ou igual a R\$ 600,00	isento
maior que R\$ 600,00 e menor ou igual a R\$ 1200,00	20%
maior que R\$ 1200,00 e menor ou igual a R\$ 2000,00	25%
maior que R\$ 2000,00	30%

```

prog se40
real salario, desconto;
imprima "\ndigite salario: ";
leia salario;
se ( salario <= 600. )
{ desconto <- 0.;}
senao
{ se ( salario <= 1200. )
{ desconto <- salario * 0.2; }
senao
{ se ( salario <= 2000. )
{ desconto <- salario * 0.25; }
senao
{ desconto <- salario * 0.30; }
}
}
imprima "\ndesconto: ",desconto;
imprima "\n";
fimprog

```

### algoritmo 130

---

Um comerciante comprou um produto e quer vendê-lo com um lucro de 45% se o valor da compra for menor que R\$ 20,00; caso contrário, o lucro será de 30%. Entrar com o valor do produto e imprimir o valor da venda.

```

prog se41
real valor;
imprima "\ndigite valor do produto: ";
leia valor;
se( valor < 20. )
{ imprima "\nValor de venda: ", valor * 1.45 ; }
senao
{ imprima "\nValor de venda: " , valor *1.3; }
imprima "\n";
fimprog

```

### algoritmo 131

---

A turma de Programação I, por ter muitos alunos, será dividida em dias de provas. Após um estudo feito pelo coordenador, decidiu-se dividi-la em três grupos. Fazer um algoritmo que leia o nome do aluno e indicar a sala em que ele deverá fazer as provas, tendo em vista a tabela a seguir e sabendo-se que todas as salas se encontram no bloco F:

*A – K : sala 101*

*L – N : sala 102*

*O – Z : sala 103*

```

prog se42
    string nome,L;
    imprima "\nDigite nome: ";
    leia nome;
    L<-strprim(nome);
    se ((L>="A" && L<="K") || (L>="a" && L<="k"))
    { imprima "\n", nome, " sala: 101"; }
    senao
    {
        se ((L>="L" && L<="N") || (L>="l" && L<="n"))
        { imprima "\n", nome, " sala: 102"; }
        senao
        {
            se ((L>="O" && L<="Z") || (L>="o" && L<="z"))
            { imprima "\n", nome, " sala: 103"; }
            senao
            { imprima "\n NOME INVALIDO"; }
        }
    }
    imprima "\n";
fimprog

```

### algoritmo 132

---

*Fazer um algoritmo que possa converter uma determinada quantia dada em reais para uma das seguintes moedas:*

- *f – franco suíço*
- *l – libra esterlina*
- *d – dólar*
- *m – marco alemão*

```

prog se43
    real r,l,d,f,m;
    string moeda;
    imprima "\ndigite valor em reais a ser convertido: ";
    leia r;
    imprima "\ndigite f - franco suico l - libra esterlina d - dolar m - marco alemão: ";
    leia moeda;
    se( moeda == "f" || moeda == "F")
    { imprima "\ndigite valor de um franco suico em reais: ";
        leia f;
        imprima "\nTotal de francos suicos: ", r /f;};
    senao
    { se( moeda == "l" || moeda == "L")
        { imprima "\ndigite valor de uma libra esterlina em reais: ";
            leia l;
    }
}

```

```

imprima "\nTotal de libras esterlinas: ", r /l;
senao
{ se( moeda == "d" || moeda == "D")
{ imprima "\ndigite valor de um dolar em reais: ";
leia d;
imprima "\nTotal de dolares: ", r /d;}
senao
{ se( moeda == "m" || moeda == "M")
{ imprima "\ndigite valor de um marco alemão em reais: ";
leia m;
imprima "\nTotal de marcos alemaes: ", r /m;}
senao
{ imprima "\nmoeda desconhecida";}
}
}
imprima "\n";
fimprog

```

### algoritmo 133

---

Segundo uma tabela médica, o peso ideal está relacionado com a altura e o sexo. Fazer um algoritmo que receba a altura e o sexo de uma pessoa, calcular e imprimir o seu peso ideal, utilizando as seguintes fórmulas:

- para homens:  $(72.7 * H) - 58$
- para mulheres:  $(62.1 * H) - 44.7$

```

prog se44
string sexo;
real h, peso;
imprima "\nentre com a altura: ";
leia h;
imprima "\nentre com o sexo M / F: ";
leia sexo;
se( sexo == "M" || sexo == "m")
{peso <- 72.7 * h - 58;}
senao
{peso <- 62.1 * h - 44.7;}
imprima "\nseu peso ideal : ", peso;
imprima "\n";
fimprog

```

### algoritmo 134

---

A confederação brasileira de natação irá promover eliminatórias para o próximo mundial. Fazer um algoritmo que receba a idade de um nadador e imprimir a sua categoria segundo a tabela a seguir:

Categoria	Idade
Infantil A	5 – 7 anos
Infantil B	8 – 10 anos
Juvenil A	11 – 13 anos
Juvenil B	14 – 17 anos
Sênior	maiores de 18 anos

```

prog se45
int idade;
imprima "\nentre com a idade: ";
leia idade;
se( idade < 5 )
{ imprima "\nnao existe categoria para essa idade" ;}
senao
{ se( idade <= 7 )
{ imprima "\ncategoria Infantil A" ;}
senao
{ se( idade <= 10 )
{ imprima "\ncategoria Infantil B";}
senao
{ se( idade <= 13 )
{ imprima "\ncategoria Juvenil A"; }
senao
{ se( idade <= 17 )
{ imprima "\ncategoria Juvenil B";}
senao
{ imprima "\ncategoria Senior";}
}
}
}
imprima "\n";
fimprog

```

### algoritmo 135

*Criar um algoritmo que leia a idade de uma pessoa e informar a sua classe eleitoral:*

- *não-eleitor (abaixo de 16 anos)*
- *eleitor obrigatório ( entre 18 e 65 anos)*
- *eleitor facultativo ( entre 16 e 18 anos e maior de 65 anos)*

```

prog se46
int idade;
imprima "\ndigite idade: ";

```

```

leia idade;
se ( idade < 16 )
{ imprima "\nNao eleitor"; }
senao
{
  se ( idade > 65 )
  { imprima "\nEleitor facultativo"; }
  senao
  { imprima "\nEleitor obrigatorio"; }
}
imprima "\n";
fimprog

```

### algoritmo 136

---

*Depois da liberação do governo para as mensalidades dos planos de saúde, as pessoas começaram a fazer pesquisas para descobrir um bom plano, não muito caro. Um vendedor de um plano de saúde apresentou a tabela a seguir. Criar um algoritmo que entre com o nome e a idade de uma pessoa e imprimir o nome e o valor que ela deverá pagar.*

- até 10 anos – R\$ 30,00
- acima de 10 até 29 anos - R\$ 60,00
- acima de 29 até 45 anos - R\$ 120,00
- acima de 45 até 59 anos - R\$ 150,00
- acima de 59 até 65 anos - R\$ 250,00
- maior que 65 anos – R\$ 400,00

```

prog se47
int idade;
string nome;
imprima "\nEnter com nome: ";
leia nome;
imprima "\nEnter com a idade: ";
leia idade;
se ( idade <= 10 )
{ imprima "\n", nome, " pagara R$ 30,00"; }
senao
{
  se ( idade <= 29 )
  { imprima "\n", nome, " pagara R$ 60,00"; }
  senao
  {
    se ( idade <= 45 )
    { imprima "\n", nome, " pagara R$ 120,00"; }
    senao
  }
}

```

```

{
    se ( idade <= 59 )
    { imprima "\n", nome, " pagara R$ 150,00"; }
    senao
    {
        se ( idade <= 65 )
        { imprima "\n", nome, " pagara R$ 250,00"; }
        senao
        { imprima "\n", nome, " pagara R$ 400,00"; }
    }
}
}
imprima "\n";
fimprog

```

### algoritmo 137

---

Ler três valores inteiros (variáveis  $a$ ,  $b$  e  $c$ ) e efetuar o cálculo da equação de segundo grau, apresentando: as duas raízes, se para os valores informados for possível fazer o cálculo (delta positivo ou zero); a mensagem "Não há raízes reais", se não for possível fazer o cálculo (delta negativo); e a mensagem "Não é equação do segundo grau", se o valor de  $a$  for igual a zero.

```

prog se48
real a, b, c, d, x1, x2;
imprima "\nEntre com valor de a: ";
leia a;
imprima "\nEntre com valor de b: ";
leia b;
imprima "\nEntre com valor de c: ";
leia c;
se (a == 0.)
{ imprima "\n Nao e equacao do 2 grau";}
senao
{
    d<- b**2 - 4 * a * c;
    se (d >=0.)
    {
        d<- raiz(d);
        x1<- (-b + d)/ (2 *a);
        x2 <-(-b - d)/ (2 *a);
        imprima "\n X1= ", x1, "\t\tX2= ", x2;
    }
    senao
    { imprima "\nNao ha raizes reais"; }
}
imprima "\n";
fimprog

```

## algoritmo 138

---

Ler um número inteiro entre 1 e 12 e escrever o mês correspondente. Caso o usuário digite um número fora desse intervalo, deverá aparecer uma mensagem informando que não existe mês com este número.

```
prog se49
int num;
imprima "\ndigite vumnumero de 1 - 12: ";
leia num ;
se (num==1)
{ imprima "\njaneiro"; }
senao
{ se (num==2)
{ imprima "\nfevereiro"; }
senao
{ se (num==3)
{ imprima "\nmarco"; }
senao
{ se (num==4)
{ imprima "\nbril"; }
senao
{ se (num==5)
{ imprima "\nmaio"; }
senao
{ se (num==6)
{ imprima "\njunho"; }
senao
{ se (num==7)
{ imprima "\njulho"; }
senao
{ se (num==8)
{ imprima "\nagosto"; }
senao
{ se (num==9)
{ imprima "\nsetembro"; }
senao
{ se (num==10)
{ imprima "\noutubro"; }
senao
{ se (num==11)
{ imprima "\nnovembro"; }
senao
{ se (num==12)
{ imprima "\ndezenbro"; }
senao
{ imprima "\nnao existe mes correspondente"; }
```

```

        }
    }
}
}

imprima "\n";
fimprog

prog se49comescolha
# nao disponivel nesta versao
int num;
imprima "\ndigite vumnumero de 1 - 12: ";
leia num ;
escolha (num)
{
    caso 1: imprima "\njaneiro"; pare;
    caso 2: imprima "\nfevereiro"; pare;
    caso 3: imprima "\nmarço"; pare;
    caso 4: imprima "\nbril"; pare;
    caso 5: imprima "\nmaio"; pare;
    caso 6: imprima "\njunho"; pare;
    caso 7: imprima "\n julho"; pare;
    caso 8: imprima "\nagosto"; pare;
    caso 9: imprima "\nsetembro"; pare;
    caso 10: imprima "\noutubro"; pare;
    caso 11: imprima "\nnovembro"; pare;
    caso12: imprima "\ndezenbro"; pare;
    senao: imprima "\nnao existe mes correspondente a este numero";
}
imprima "\n";
fimprog

```

### algoritmo 139

*Sabendo que somente os municípios que possuem mais de 20.000 eleitores aptos têm segundo turno nas eleições para prefeito caso o primeiro colocado não tenha mais do que 50% dos votos, fazer um algoritmo que leia o nome do município, a quantidade de eleitores aptos, o número de votos do candidato mais votado e informar se ele terá ou não segundo turno em sua eleição municipal.*

```

prog se50
int ne, votos;

```

```

string nome;
imprima "\nDigite nome do Municipio: ";
leia nome;
imprima "\nnumero de eleitores aptos: ";
leia ne;
imprima "\nnumero de votos do candidato mais votado: ";
leia votos;
se (ne >20000 && votos <= ne div 2)
{ imprima "\n", nome, " tera segundo turno"; }
senao
{ imprima "\n", nome, " nao terá segundo turno"; }
imprima "\n";
fimprog

```

## algoritmo 140

---

*Um restaurante faz uma promoção semanal de descontos para clientes de acordo com as iniciais do nome da pessoa. Criar um algoritmo que leia o primeiro nome do cliente, o valor de sua conta e se o nome iniciar com as letras A, D, M ou S, dar um desconto de 30%. Para o cliente cujo nome não se inicia por nenhuma dessas letras, exibir a mensagem “Que pena. Nesta semana o desconto não é para seu nome; mas continue nos prestigiando que sua vez chegará”.*

```

prog se51
string nome,L;
real vc, vcd;
imprima "\nDigite nome: ";
leia nome;
imprima "\nDigite valor da conta: ";
leia vc;
L<-strprim(nome);
se( L=="A" || L=="a" || L=="D" || L=="d" || L=="M" || L=="m" || L=="S"
|| L=="s" )
{ imprima "\n", nome, " valor da conta com desconto: R$ ", formatar(vc *
0.7, 2); }
senao
{ imprima "\n Que pena. Nesta semana o desconto não é para seu nome; mas
continue nos prestigiando que sua vez chegará"; }
imprima "\n";
fimprog

```

## algoritmo 141

---

*Criar um algoritmo que leia o nome e o total de pontos de três finalistas de um campeonato de pingue-pongue e exibir a colocação da seguinte forma:*

Vencedor:	XXXX pts
Segundo colocado:	XXXX pts
Terceiro colocado:	XXXX pts

```

prog se52
    int p1, p2, p3, aux;
    string n1,n2,n3,auxn;
    imprima "\ndigite 1 nome: ";
    leia n1;
    imprima "\ndigite pontos: ";
    leia p1;
    imprima "\ndigite 2 nome: ";
    leia n2;
    imprima "\ndigite pontos: ";
    leia p2;
    imprima "\ndigite 3 nome: ";
    leia n3;
    imprima "\ndigite pontos: ";
    leia p3;
    se ( p1 < p2 )
    { aux <- p1; p1 <- p2; p2 <- aux; auxn <- n1; n1 <- n2; n2 <- auxn; }
    se ( p1 < p3 )
    { aux <- p1; p1 <- p3; p3 <- aux; auxn <- n1; n1 <- n3; n3 <- auxn; }
    se ( p2 < p3 )
    { aux <- p2; p2 <- p3; p3 <- aux; auxn <- n2; n2 <- n3; n3 <- auxn; }
    imprima "\nVENCEDOR      : ", n1, " ",p1, " pontos";
    imprima "\nSEGUNDO COLOCADO : ", n2, " ",p2, " pontos";
    imprima "\nTERCEIRO COLOCADO: ", n3, " ",p3, " pontos";
    imprima "\n";
fimprog

```

## algoritmo 142

---

*Em um campeonato nacional de arco-e-flecha, tem-se equipes de três jogadores para cada estado. Sabendo-se que os arqueiros de uma equipe não obtiveram o mesmo número de pontos, criar um algoritmo que informe se uma equipe foi classificada, de acordo com a seguinte especificação:*

- ler os pontos obtidos por cada jogador da equipe;
- mostrar esses valores em ordem decrescente;
- se a soma dos pontos for maior do que 100, imprimir a média aritmética entre eles; senão, imprimir a mensagem "Equipe desclassificada".

```

prog se53
    int p1, p2, p3, aux, soma;
    imprima "\n Digite os pontos do primeiro jogador: ";
    leia p1;
    imprima "\n Digite os pontos do segundo jogador: ";
    leia p2;

```

```

imprima "\n Digite os pontos do terceiro jogador: ";
leia p3;
se(p1 > p2)
{ aux<- p1 ; p1<- p2; p2<- aux; }
se(p1 > p3)
{ aux<- p1 ; p1<- p3; p3<- aux; }
se(p2 > p3)
{ aux<- p2 ; p2<- p3; p3<- aux; }
imprima "\n p1=", p1, "\n p2=", p2, "\n p3=", p3;
soma <- p1 + p2 + p3;
se(soma > 100)
{ imprima "\nMedia=", formatar(soma/3,2);}
senao
{ imprima "\nEquipe desclassificada!"}
imprima "\n";
fimprog

```

### algoritmo 143

---

*Criar um algoritmo que verifique a(s) letra(s) central(is) de uma palavra. Se o número de caracteres for ímpar, ele verifica se a letra central é uma vogal; caso contrário, verifica se é um dos dígrafos rr ou ss (só precisa testar letras minúsculas).*

```

prog se54
string pal, p1, p2, p3;
int n1, n2;
imprima "\ndigite uma palavra: ";
leia pal;
se(strtam(pal) % 2 == 0)
{ n1<-strtam(pal) div 2;
  n2<- n1-1;
  p1<-strelem(pal,n1);
  p2<-strelem(pal,n2);
  p3<-strconcat(p2,p1);
  se( p3 == "rr" || p3 == "ss" )
  {imprima "\nE rr OU ss";}
  senao
  {imprima "\nNAO E rr OU ss";}
}
senao
{ n1<-strtam(pal) div 2;
  p1<-strelem(pal,n1);
  se( p1 == "a" || p1 == "e"|| p1 == "i" || p1 == "o" || p1 == "u")
  {imprima "\nVOGAL";}
  senao
  {imprima "\nNAO E VOGAL";}
}
imprima "\n";
fimprog

```

## algoritmo 144

---

O banco XXX concederá um crédito especial com juros de 2% aos seus clientes de acordo com o saldo médio no último ano. Fazer um algoritmo que leia o saldo médio de um cliente e calcule o valor do crédito de acordo com a tabela a seguir. Imprimir uma mensagem informando o saldo médio e o valor do crédito.

SALDO MÉDIO	PERCENTUAL
de 0 a 500	nenhum crédito
de 501 a 1000	30% do valor do saldo médio
de 1001 a 3000	40% do valor do saldo médio
acima de 3001	50% do valor do saldo médio

```
prog se55
real saldomedio, credito;
imprima "\ndigite saldo medio: ";
leia saldomedio;
se( saldomedio < 501. )
{ credito <-0.; }
senao
{ se(saldomedio < 1001.)
{ credito <- saldomedio * 0.3;}
senao
{ se(saldomedio < 3001.)
{ credito <- saldomedio * 0.4;}
senao
{ credito <- saldomedio * 0.5;}
}
}
se(credito <> 0.)
{ imprima "\nComo seu saldo era de:",saldomedio," seu credito sera
de:",credito;}# continuacao da linha anterior
senao
{ imprima "\nComo seu saldo era de:", saldomedio,", voce nao tera
nenhum credito";}# continuacao da linha anterior
imprima "\n";
fimprog
```

## algoritmo 145

---

A biblioteca de uma universidade deseja fazer um algoritmo que leia o nome do livro que será emprestado, o tipo de usuário (professor ou aluno) e possa imprimir um recibo conforme mostrado a seguir. Considerar que o professor tem dez dias para devolver o livro e o aluno só três dias.

Nome do livro

Tipo de usuário:

110 | Total de Dias:

```

prog se56
string livro, tipo;
int dia;
imprima "\ndigite nome do livro: ";
leia livro;
imprima "\ndigite: professor / aluno: ";
leia tipo;
se( tipo == "professor" || tipo == "Professor" || tipo == "PROFESSOR")
{ dia <-10 ;}
senao
{ se( tipo == "aluno" || tipo == "Aluno" || tipo == "ALUNO")
  { dia <-3 ;}
  senao
  { dia <- 0;}
}
se( dia == 0)
{ imprima "\n Tipo de usuario desconhecido";}
senao
{ imprima "\n nome do livro: ", livro;
  imprima "\n nome tipo de usuario: ", tipo;
  imprima "\n total de dias: ", dia;
}
imprima "\n";
fimprog

```

## algoritmo 146

---

Fazer um algoritmo que leia o percurso em quilômetros, o tipo do carro e informe o consumo estimado de combustível, sabendo-se que um carro tipo C faz 12 km com um litro de gasolina, um tipo B faz 9 km e o tipo C, 8 km por litro.

```

prog se57
real percurso, consumo;
string tipo;
imprima "\ndigite tipo de carro( A / B / C): ";
leia tipo;
imprima "\ndigite percurso: ";
leia percurso;
se ( tipo == "C" || tipo == "c")
{ consumo <- percurso / 12;}
senao
{
  se ( tipo == "B" || tipo == "b")
  { consumo <- percurso / 10; }
  senao
  {
    se ( tipo == "A" || tipo == "a")

```

```

{ consumo <- percurso / 8; }
senao
{ consumo <- 0.; }
}
}
se ( consumo < > 0.)
{ imprima "\nConsumo estimado em litros: ", formatar(consumo,2);}
senao
{ imprima "\nModelo inexistente";}
imprima "\n";
fimprog

```

### algoritmo 147

---

*Criar um algoritmo que informe a quantidade total de calorias de uma refeição a partir da escolha do usuário que deverá informar o prato, a sobremesa e bebida (veja a tabela a seguir).*

PRATO	SOBREMESA	BEBIDA
Vegetariano 180cal	Abacaxi 75cal	Chá 20cal
Peixe 230cal	Sorvete diet 110cal	Suco de laranja 70cal
Frango 250cal	Mousse diet 170cal	Suco de melão 100cal
Carne 350cal	Mousse chocolate 200cal	Refrigerante diet 65cal

```

prog se58
int op, os, ob, calorias;
calorias <- 0;
imprima "\nDigite 1- Vegetariano 2- Peixe 3- Frango 4- Carne: ";
leia op;
se(op==1)
{calorias <- calorias +180;}
senao
{ se(op==2)
{calorias <- calorias +230;}
senao
{se(op==3)
{calorias <- calorias +250;}
senao
{se(op==4)
{calorias <- calorias +350;}
}
}
}
imprima "\nDigite 1- Abacaxi 2- Sorvete diet 3- Suco melao 4-
refrigerante diet: ";
leia op;

```

```

se(op==1)
{calorias <- calorias +75;}
senao
{ se(op==2)
{calorias <- calorias +110;}
senao
{se(op==3)
{calorias <- calorias +170;}
senao
{se(op==4)
{calorias <- calorias +200;}
}
}
}
imprima "\nDigite 1- Cha  2- Suco de laranja  3- Mousse diet  4- Mousse
de chocolate: ";
leia op;
se(op==1)
{calorias <- calorias +20;}
senao
{ se(op==2)
{calorias <- calorias +70;}
senao
{se(op==3)
{calorias <- calorias +100;}
senao
{se(op==4)
{calorias <- calorias +65;}
}
}
}
imprima "\n Total de calorias: ", calorias;
imprima "\n";
fimprog

```

### **algoritmo 148**

---

*Criar um algoritmo que leia o destino do passageiro, se a viagem inclui retorno (ida e volta) e informar o preço da passagem conforme a tabela a seguir.*

<b>DESTINO</b>	<b>IDA</b>	<b>IDA E VOLTA</b>
Região Norte	R\$500,00	R\$900,00
Região Nordeste	R\$350,00	R\$650,00
Região Centro-Oeste	R\$350,00	R\$600,00
Região Sul	R\$300,00	R\$550,00

```

prog se59
    int op, iv;
    imprima "\nDigite 1- Regiao Norte 2- Regiao Nordeste 3- Regiao Centro-Oeste 4-
    Regiao Sul: ";
    leia op;
    imprima "\nDigite 1- Ida 2- Ida e Volta: ";
    leia iv;
    se(op == 1)
    { se(iv == 1)
        { imprima "\nO valor da passagem de ida para R.Norte R$500.00";}
        senao
        { imprima "\nO valor da passagem de ida-volta para R.Norte: R$950.00";}
    }
    senao
    { se(op == 2)
        { se(iv == 1)
            { imprima "\nO valor da passagem de ida para R.Nordeste: R$350.00";}
            senao
            { imprima "\nO valor da passagem de ida-volta para R.Nordeste: R$650.00";}
        }
        senao
        { se(op == 3)
            { se(iv == 1)
                { imprima "\nO valor da passagem de ida para R.C.Oeste: R$350.00";}
                senao
                { imprima "\nO valor da passagem de ida-volta para R.C.Oeste: R$600.00";}
            }
            senao
            { se(op == 4)
                { se(iv == 1)
                    { imprima "\nO valor da passagem de ida para R.Sul: R$300.00";}
                    senao
                    { imprima "\nO valor da passagem de ida-volta para R.Sul: R$550.00";}
                }
                senao
                { imprima "\nOpcao Inexistente";}
            }
        }
    }
    imprima "\n";
fimprog

```

### algoritmo 149

Um comerciante calcula o valor da venda, tendo em vista a tabela a seguir:

<b>VALOR DA COMPRA</b>	<b>VALOR DA VENDA</b>
<i>Valor &lt; R\$ 10,00</i>	<i>lucro de 70%</i>
<i>R\$10,00 ≤ valor &lt; R\$ 30,00</i>	<i>lucro de 50%</i>
<i>R\$30,00 ≤ valor &lt; R\$ 50,00</i>	<i>lucro de 40%</i>
<i>Valor ≥ R\$50,00</i>	<i>lucro de 30%</i>

*Criar o algoritmo que possa entrar com nome do produto e valor da compra e imprimir o nome do produto e o valor da venda.*

```
prog se60
real vcompra, vvenda;
string nomep;
imprima "\ndigite nome do produto: ";
leia nomep;
imprima "\ndigite valor da compra: ";
leia vcompra;
se ( vcompra < 10. )
{ vvenda <- vcompra * 1.7; }
senao
{
se ( vcompra < 30. )
{ vvenda <- vcompra * 1.5; }
senao
{ vvenda <- vcompra * 1.3; }
}
}
imprima "\n", nomep, "será vendido por R$\"", formatar(vvenda +
0.001,2);
imprima "\n";
fimprog
```

## algoritmo 150

---

*Criar um algoritmo que leia um ângulo em graus e apresente:*

- o seno do ângulo, se o ângulo pertencer a um quadrante par;
- o co-seno do ângulo, se o ângulo pertencer a um quadrante ímpar.

```
prog se61
real ang, rang;
imprima "\ndigite angulo em graus: ";
leia ang;
rang <- ang * pi / 180 ;
se((rang > pi/2 && rang <= pi) || (rang > 3*pi/2 && rang <= 2*pi))
{ imprima "\nseno: ", sen(rang); }
senao
{ imprima "\nco-seno: ", cos(rang); }
imprima "\n";
fimprog
```

## algoritmo 151

Um endocrinologista deseja controlar a saúde de seus pacientes e, para isso, se utiliza do Índice de Massa Corporal (IMC). Sabendo-se que o IMC é calculado através da seguinte fórmula:

$$IMC = \frac{\text{peso}}{\text{altura}^2}$$

Onde:

- peso é dado em Kg
- altura é dada em metros

Criar um algoritmo que apresente o nome do paciente e sua faixa de risco, baseando-se na seguinte tabela:

IMC	FAIXA DE RISCO
abaixo de 20	abaixo do peso
a partir de 20 até 25	normal
acima de 25 até 30	excesso de peso
acima de 30 até 35	obesidade
acima de 35	obesidade mórbida

```
prog se62
    real peso, altura, imc;
    imprima "\ndigite peso: ";
    leia peso;
    imprima "\ndigite altura: ";
    leia altura;
    imc <- peso / altura **2;
    se ( imc < 20. )
    {   imprima "\nabaixo do peso"; }
    senao
    {
        se ( imc <=25. )
        {   imprima "\nnormal"; }
        senao
        {
            se ( imc <=30. )
            {   imprima "\nexcesso de peso "; }
            senao
            {
                se ( imc <=35. )
                {   imprima "\nobesidade "; }
                senao
                {   imprima "\no obesidade mórbida"; }
            }
        }
    }
    imprima "\n";
fimprog
```

## algoritmo 152

Criar um algoritmo que a partir da idade e peso do paciente calcule a dosagem de determinado medicamento e imprima a receita informando quantas gotas do medicamento o paciente deve tomar por dose. Considere que o medicamento em questão possui 500 mg por ml, e que cada ml corresponde a 20 gotas.

■ Adultos ou adolescentes desde 12 anos, inclusive, se tiverem peso igual ou acima de 60 quilos devem tomar 1000 mg; com peso abaixo de 60 quilos devem tomar 875 mg.

■ Para crianças e adolescentes abaixo de 12 anos é dosagem é calculada pelo peso corpóreo conforme a tabela a seguir:

5 kg a 9 kg = 125 mg	24.1 kg a 30 kg = 500 mg
9.1 kg a 16 kg = 250 mg	acima de 30 kg = 750 mg
16.1 kg a 24 kg = 375 mg	

```
prog se63
  int idade;
  real peso,gotas;
  imprima "\ndigite peso: ";
  leia peso;
  imprima "\ndigite idade: ";
  leia idade;
  gotas <- 500 / 20; # calculo do número de mg por gotas
  se (peso <5.)
  { imprima "\nnao pode tomar o remedio porque nao tem peso. Consulte
medico."; }
  senao
  {
    se ( idade >= 12)
    {
      se ( peso >= 60. )
      { imprima "\nTomar ",1000 / gotas, " gotas"; }
      senao
      { imprima "\nTomar ", 875 /gotas, " gotas"; }
    }
    senao
    {
      se ( peso <= 9.)
      { imprima "\nTomar ", 125 / gotas, " gotas"; }
      senao
      {
        se ( peso <= 16. )
        { imprima "\nTomar ", 250 / gotas, " gotas"; }
        senao
        {
```

```

    se ( peso <= 24. )
    { imprima "\nTomar ", 375 / gotas, " gotas"; }
    senao
    {
        se ( peso <= 30. )
        { imprima "\nTomar ", 500 / gotas, " gotas"; }
        senao
        { imprima "\nTomar ", 750 / gotas, " gotas"; }
    }
}
imprima "\n";
fimprog

```

### algoritmo 153

---

O prefeito do Rio de Janeiro contratou uma firma especializada para manter os níveis de poluição considerados ideiais para um país do 1º mundo. As indústrias, maiores responsáveis pela poluição, foram classificadas em três grupos. Sabendo-se que a escala utilizada varia de 0,05 e que o índice de poluição aceitável é até 0,25, fazer um algoritmo que possa imprimir intimações de acordo com o índice a seguir:

índice	indústrias que receberão intimação
0,3	1º grupo
0,4	1º e 2º grupos
0,5	1º, 2º e 3º grupos

```

prog se64
    real indice;
    imprima "\ndigite indice de poluicao: ";
    leia indice;
    se ( indice >= 0.5 )
    { imprima "\nsuspender atividades as industrias dos grupos 1, 2 e 3"; }
    senao
    {
        se ( indice >= 0.4 )
        { imprima "\nsuspender atividades as industrias dos grupos 1 e 2 "; }
        senao
        {
            se ( indice >= 0.3 )
            { imprima "\nsuspender atividades as industrias dos grupos 1 "; }
            senao
            { imprima "\nindice de poluicao aceitavel para todos os grupos "; }
        }
    }
    imprima "\n";
fimprog

```

## algoritmo 154

A polícia rodoviária resolveu fazer cumprir a lei e cobrar dos motoristas o DUT. Sabendo-se que o mês em que o emplacamento do carro deve ser renovado é determinado pelo último número da placa do mesmo, criar um algoritmo que, a partir da leitura da placa do carro, informe o mês em que o emplacamento deve ser renovado.

↳ Leia a placa do carro em uma variável caracter.

```
prog se65
    string placa, p;
    imprima "\ndigite placa: ";
    leia  placa ;
    p <- strult(placa);
    se ( p == "1" )
        { imprima "\njaneiro"; }
    senao
    { se ( p == "2" )
        { imprima "\nfevereiro"; }
    senao
    { se ( p == "3" )
        { imprima "\nmarco"; }
    senao
    { se ( p == "4" )
        { imprima "\nabril"; }
    senao
    { se ( p == "5" )
        { imprima "\nmaio"; }
    senao
    { se ( p == "6" )
        { imprima "\njunho"; }
    senao
    { se ( p == "7" )
        { imprima "\njulho"; }
    senao
    { se ( p == "8" )
        { imprima "\nagosto"; }
    senao
    { se ( p == "9" )
        { imprima "\nsetembro"; }
    senao
    { se ( p == "0" )
        { imprima "\noutubro"; }
    senao
        { imprima "\nopcao invalida"; }
```

## algoritmo 155

Ler uma palavra e, se ela começar pela letra L ou D (também deve ser considerado l ou d), formar uma nova palavra que terá os dois primeiros caracteres e o último, caso contrário a nova palavra será formada por todos os caracteres menos o primeiro.

```
prog se66
  string pal, pall, pal2, letra;
  int tam;
  imprima "\ndigite palavra: ";
  leia pal;
  letra <- strprim(pal);
  se( letra == "l" || letra == "L" || letra == "d" || letra == "D" )
  { pall <- strnprim(pal,2);
    pal2 <- strlult(pal);
    pall <- strconcat(pall,pal2);}
  senao
  {pall <- strresto(pal);}
  imprima "\npalavra: ", pall;
  imprima "\n";
fimprog
```

## algoritmo 156

*Criar um algoritmo que leia uma data (dia, mês e ano em separado) e imprima se a data é válida ou não.*

```
prog se67
    int ANO, MES, DIA, VD;
    imprima "\ndigite dia: ";
    leia DIA;
    imprima "\ndigite mes: ";
    leia MES;
    imprima "\ndigite ano: ";
    leia ANO;
    se(ANO>=1)
```

```

{ VD <- 1;
se( MES < 1 || MES > 12 || DIA < 1 || DIA > 31)
{ VD <- 0; }
senao
{
  se( (MES == 4 || MES == 6 || MES == 9 || MES == 11) && DIA > 30)
  {VD <- 0;}
  senao
  { se(MES == 2)
    {
      se((ANO % 4==0 && ANO % 100 < > 0) || ANO % 400==0)
      {
        se(DIA > 29)
        { VD <- 0;}
      }
      senao
      {
        se( DIA > 28)
        { VD <- 0;}
      }
    }
  }
}
senao
{ VD <- 0;}
se( VD == 0 )
{ imprima "\nDATA INVALIDA";}
senao
{ imprima "\nDATA VALIDA";}
imprima "\n";
fimprog

```

### algoritmo 157

Criar um algoritmo que leia uma data (no formato ddmmaaaa) e imprimir se a data é válida ou não.

```

prog se68
int DATA, ANO, MES, DIA, VD;
imprima "\ndata no formato ddmmaaaa: ";
leia DATA;
DIA <- DATA div 1000000;
MES <- DATA % 1000000 div 10000;
ANO <- DATA % 10000;
se(ANO >=1)
{ VD <- 1;
  se(MES > 12 || DIA < 1 || DIA > 31)
  { VD <- 0; }
}

```

```

senao
{
    se((MES == 4||MES == 6||MES == 9||MES == 11)&& DIA > 30)
    {VD <- 0;}
    senao
    {
        se(MES == 2)
        {
            se((ANO % 4==0 && ANO % 100 < > 0) || ANO % 400==0)
            {
                se(DIA > 29)
                { VD <- 0;}
            }
            senao
            { se( DIA > 28)
            { VD <- 0;}
            }
        }
    }
}
senao
{ VD <- 0;}
se( VD == 0 )
{imprima "\nDATA INVALIDA";}
senao
{imprima "\nDATA VALIDA";}
imprima "\n";
fimprog

```

### algoritmo 158

---

*Criar um algoritmo que entre com o valor de x, calcule e imprima o valor de f(x).*

$$f(x) = \frac{8}{2-x}$$

```

prog se69
real x, fx;
imprima "\ndigite valor de x: ";
leia x;
se(x < > 2. )
{fx <- 8 / (2 - x);
imprima "\nf(x)= ",fx;
}
senao
{imprima "\nNAO PODE SER FEITA";}
imprima "\n";
fimprog

```

## algoritmo 159

Criar um algoritmo que entre com o valor de  $x$ , calcule e imprima o valor de  $f(x)$ .

$$f(x) = \frac{5x + 3}{\sqrt{x^2 - 16}}$$

```
prog se70
    real x, fx;
    imprima "\ndigite valor de x: ";
    leia x;
    se(x > 4. || x < (-4.))
    {fx <- (5 * x + 3) / raiz(x**2 - 16);
    imprima "\nf(x)= ",fx;
    }
    senao
    {imprima "\nNAO PODE SER FEITA";
    imprima "\n";
    fimprog
```

## algoritmo 160

Entrar com o valor de  $x$  e imprimir  $y$ :

$$Y = f(x) \begin{cases} 1, & \text{se } x \leq 1 \\ 2, & \text{se } 1 < x \leq 2 \\ x^2, & \text{se } 2 < x \leq 3 \\ x^3, & \text{se } x > 3 \end{cases}$$

```
prog se71
    real x, y;
    imprima "\ndigite valor de x: ";
    leia x;
    se(x <= 1.)
    {y <- 1.;}
    senao
    { se(x <= 2.)
    {y <- 2.;}
    senao
    { se(x <= 3.)
    {y <- x **2;}
    senao
    {y <- x **3;}
    }
    }
    imprima "\nvalor de f(x): ",y;
    imprima "\n";
    fimprog
```



## Capítulo 4

# Estruturas de repetição: para, enquanto e faca-enquanto

Este capítulo é um dos mais importantes do livro. A partir desse ponto, os alunos começam a ter muita dificuldade na resolução dos algoritmos por não entenderem onde as estruturas de repetição deverão ser colocadas para atender às exigências dos enunciados. Uma das perguntas mais comuns é: “o comando **leia** ven antes ou depois da estrutura de repetição?” Leia com muita atenção!

Nossos algoritmos precisarão ser executados mais de uma vez e, para que não tenhamos de reescrever trechos idênticos que aumentariam consideravelmente o tamanho, utilizamos três estruturas de repetição.

### ESTRUTURA DO PARA

Usada quando o número de repetições for conhecido durante a elaboração do algoritmo ou quando puder ser fornecido durante a execução.

Ex: Criar um algoritmo que possa entrar com a nota de 20 alunos e imprimir a média da turma.

```
para ( valor inicial ; <condição>; <valor do incremento> )  
{  
    bloco de comandos  
}
```

< valor inicial >; – é uma expressão do tipo:

```
<identificador> <- <valor inicial> ;
```

Exemplos:

- |                      |  |
|----------------------|--|
| a <- 0;              | A variável a recebe o valor inicial 0.   |
| c <- 100;            | A variável c recebe o valor inicial 100.   |
| x <- strtam(pal) -1; | A variável x recebe o valor inicial que é igual ao número de caracteres da variável pal menos 1. |

< condição >; – é uma expressão do tipo:

```
<identificador> <=, <, > ou >= <valor final> ;
```

Exemplos:

- |          |  |
|----------|--|
| a <= 10; | A variável a poderá receber valores enquanto forem menores ou iguais a 10. |
| c >= 2;  | A variável c poderá receber valores enquanto forem maiores ou iguais a 2.  |
| x >= 0;  | A variável x poderá receber valores enquanto forem maiores ou iguais a 0.  |

< valor do incremento > - é uma expressão do tipo:

```
<identificador> <- <identificador> + ou - valor
```

Exemplos:

- |             |  |
|-------------|--|
| a <- a + 1; | A variável a é incrementada de 1. /* contador */ |
| c <- c - 2; | A variável c é decrementada de 2.                |
| x <- x - 1; | A variável x é decrementada de 1.                |

Já vimos que os operadores `++` e `--` são equivalentes aos comandos de atribuição quando a variável é incrementada ou decrementada de 1; por isso, o primeiro e o terceiro exemplos poderiam ser escritos assim:

- |       |                                   |
|-------|-----------------------------------|
| a ++; | A variável a é incrementada em 1. |
| x --; | A variável x é decrementada de 1. |

Observações

1. O identificador tem de ser do tipo int.

2. Os valores inicial e final poderão ser constantes numéricas inteiras, funções que retornem números inteiros ou expressões que retornem números inteiros.
3. O valor que é incrementado ou decrementado da variável poderá ser constante ou uma outra variável.
4. Esta estrutura é a própria expressão de uma PA, pois ela gera valores automaticamente na MP, veja bem:

<pre> para (a &lt;- 1 ; a &lt;= 5; a++)     MP         c         1         2         3         4         5         6     </pre>	<p>PA 1 : 2 : 3 : 4 : 5</p> <p>sabendo-se que a fórmula para calcular o enésimo termo de uma PA é:</p> $a_n = a_1 + (n - 1).r,$ <p>temos:</p> $a_1 = 1 \quad a_n = 5$
---	---

5. O número de repetições do bloco de comandos será igual ao NÚMERO DE TERMOS DA SÉRIE; portanto, na maioria das vezes, não importam os valores que a variável que controla a repetição assumirá.
6. A variável que controla a repetição deverá ser declarada e poderá ser impressa se precisarmos dela para numerar uma lista, posicionar etc.
7. A variável que controla a repetição JAMAIS aparecerá num comando de leitura dentro do bloco de repetição.

Vejamos alguns exemplos:

### algoritmo 161

*Criar um algoritmo que entre com cinco números e imprimir o quadrado de cada número.*

```

prog exemplo
    int c;
    real num;
    para( c<- 1; c <= 5 ; c++)
    {
        imprima "\nnumero: ";
        leia num ;
        imprima "quadrado: ", num ** 2;
    }
    imprima "\n";
fimprog

```

MP		Saída
c	num	
1	2	numero:2.
2	4	4.0
3	5	numero:4.
4	8	16.0
5	12	numero:5.
6		25.0
		numero:8.
		64.0
		numero:12.
		144.0

Vamos explicar:

1. Após serem reservadas posições na MP para as variáveis c e num, é executado o comando para.
2. O primeiro argumento c<- 1 só é executado uma vez antes do loop.
3. Os comandos imprima, leia e imprima são executados pela primeira vez.
4. Após }, o comando c++ é executado.
5. O comando c <= 5 é executado e enquanto a variável c estiver armazenando um valor menor ou igual a 5, o bloco se repete; mas, quando o valor ultrapassar 5, como no exemplo 6, o fluxo do algoritmo passa para o primeiro comando após }.

## algoritmo 162

Criar um algoritmo que imprima todos os números pares no intervalo 1-10.

```
prog exemplo
int c;
para( c<- 2; c <= 10 ; c<- c + 2)
{ imprima "\n", c; }
imprima "\n";
fimprog
```

MP	Saída
c	
2	2
4	4
6	6
8	8
10	10
12	

1. Após ser reservada uma posição na MP para a variável c, o comando para é executado.
2. O primeiro argumento c<- 2 só é executado uma vez antes do loop.
3. O comando imprima “\n”,c; é executado pela primeira vez.
4. Após }, o comando c<- c + 2 é executado.
5. O comando c <= 10 é executado e enquanto a variável c estiver armazenando um valor menor ou igual a 10, o bloco se repete; mas, quando o valor ultrapassar 10, como no exemplo 12, {o fluxo do algoritmo passa para o primeiro comando após }.
6. *Nenhuma série tem entrada de dados no bloco de repetição.*

## PARA dentro de PARA

Imagine você dando uma volta na Lagoa Rodrigo de Freitas (Rio de Janeiro): *passa uma vez pelo Corte do Cantagalo, uma vez pelo Clube do Flamengo, uma vez pelo Clube Piraquê, uma vez pelos fundos do Clube Militar*; esse trajeto representa seus algoritmos feitos até agora com os comandos aprendidos.

Imagine que seu preparo físico melhorou e agora você consegue dar três ou cinco voltas ou, quem sabe, até dez voltas na Lagoa; isso significa que você passará dez vezes pelos mesmos lugares. Esse exemplo representa a estrutura do Para.

Imagine agora que você, orientado por um profissional especializado, passou a dar três voltas na Lagoa mas, em frente ao Clube do Flamengo, você fará cinco abdominais. Se, em cada volta você faz cinco abdominais, ao final, você terá feito 15 abdominais e terá dado três voltas na Lagoa. Isso representa um Para dentro de um PARA. Acompanhe:

voltaLagoa	abdominais	
1	1	1 <sup>a</sup> volta na Lagoa
	2	1 <sup>o</sup> abdominal
	3	2 <sup>o</sup> abdominal
	4	3 <sup>o</sup> abdominal
	5	4 <sup>o</sup> abdominal
	6	5 <sup>o</sup> abdominal
		Quando ia contar o 6 <sup>o</sup> abdominal, PAROU e voltou a correr

voltaLagoa	abdominais	
2	4 2 3 4 5 6	2 <sup>a</sup> volta na Lagoa 1 <sup>o</sup> abdominal 2 <sup>o</sup> abdominal 3 <sup>o</sup> abdominal 4 <sup>o</sup> abdominal 5 <sup>o</sup> abdominal Quando ia contar o 6 <sup>o</sup> abdominal, PAROU e voltou a correr
3	4 2 3 4 5 6	3 <sup>a</sup> volta na Lagoa 1 <sup>o</sup> abdominal 2 <sup>o</sup> abdominal 3 <sup>o</sup> abdominal 4 <sup>o</sup> abdominal 5 <sup>o</sup> abdominal
3	6	Quando ia contar o 6 <sup>o</sup> abdominal, PAROU e voltou a correr
4	6	Quando ia contar a 4 <sup>a</sup> volta, PAROU e foi para casa

Vejamos como poderíamos representar em algoritmo:

### algoritmo 163

```

prog corredor
int voltaLagoa, abdominais;
para( voltaLagoa <- 1 ; voltaLagoa <=3 ; voltaLagoa++)
{ imprima "\n",voltaLagoa,"a volta na Lagoa";
  para( abdominais<- 1 ; abdominais <=5; abdominais++)
  { imprima "\n ", abdominais, "o abdominal";   }
}
imprima "\n";
fimprog

```

Você pode observar que, cada vez que chegava em frente ao Clube do Flamengo, começava a contar a partir do 1 seus abdominais e só parava quando sua próxima contagem fosse 6.

### DESAFIO

- Você já pensou em simular uma pausa usando a estrutura do *para*?

Comecei  
<após alguns segundos>  
Parei

## algoritmo 164

```
prog desafiol
    int a;
    imprima "\nComecei\n";
    para(a<-1; a<=50000;a++)
    {
    }
    imprima "\nFim";
    imprima "\n";
fimprog
```

- Você já pensou em fazer várias tabulações usando a estrutura do `para` ?

1 中国科学院植物研究所 63

## Zonal

63

Zona9

## algoritmo 165

```
prog desafio2
    int a;
    imprima "zona1\n";
    para(a<-1; a<=8; a++)
    { imprima "\t";}
    imprima "zona9";
    imprima "\n";
fimprog
```

- Você já pensou em triangular uma matriz de ordem 10 usando a estrutura do para?

## *TODOS*

1-1	1-2	1-3	1-4	1-5	1-6	1-7	1-8	1-9	1-10
2-1	2-2	2-3	2-4	2-5	2-6	2-7	2-8	2-9	2-10
3-1	3-2	3-3	3-4	3-5	3-6	3-7	3-8	3-9	3-10
4-1	4-2	4-3	4-4	4-5	4-6	4-7	4-8	4-9	4-10
5-1	5-2	5-3	5-4	5-5	5-6	5-7	5-8	5-9	5-10
6-1	6-2	6-3	6-4	6-5	6-6	6-7	6-8	6-9	6-10
7-1	7-2	7-3	7-4	7-5	7-6	7-7	7-8	7-9	7-10
8-1	8-2	8-3	8-4	8-5	8-6	8-7	8-8	8-9	8-10
9-1	9-2	9-3	9-4	9-5	9-6	9-7	9-8	9-9	9-10
10-1	10-2	10-3	10-4	10-5	10-6	10-7	10-8	10-9	10-10

## algoritmo 166

---

```
prog desafio31
int L,c,t;
imprima "\nTODOS\n";
para(L<-1;L<=10;L++)
{
    para(c<-1;c<=10;c++)
    { imprima L, "-",c, "\t" ; }
    imprima "\n";
}
imprima "\n";
fimprog
```

ACIMA DA DIAGONAL PRINCIPAL

1-2	1-3	1-4	1-5	1-6	1-7	1-8	1-9	1-10
	2-3	2-4	2-5	2-6	2-7	2-8	2-9	2-10
		3-4	3-5	3-6	3-7	3-8	3-9	3-10
			4-5	4-6	4-7	4-8	4-9	4-10
				5-6	5-7	5-8	5-9	5-10
					6-7	6-8	6-9	6-10
						7-8	7-9	7-10
							8-9	8-10
								9-10

## algoritmo 167

---

```
prog desafio32
int L,c,t;
imprima "\nACIMA DA DIAGONAL PRINCIPAL\n";
para(L<-1;L<=9;L++)
{
    para(c<-L+1;c<=10;c++)
    { imprima "\t",L, "-",c; }
    imprima "\n";
    para(t<-1;t<=L;t++)
    {imprima "\t";}
}
imprima "\n";
fimprog
```

## DIAGONAL PRINCIPAL

1-1

2-2

3-3

4-4

5-5

6-6

7-7

8-8

9-9

10-10

## algoritmo 168

```
prog desafio33
int L,c,t;
imprima "\nDIAGONAL PRINCIPAL\n";
para(L<-1;L<=10;L++)
{
    imprima L, "-", L, "\n";
    para(t<-1;t<=L;t++)
        { imprima "\t"; }
}
imprima "\n";
fimprog
```

## ABAIXO DA DIAGONAL PRINCIPAL

2-1

3-1    3-2

4-1    4-2    4-3

5-1    5-2    5-3    5-4

6-1    6-2    6-3    6-4    6-5

7-1    7-2    7-3    7-4    7-5    7-6

8-1    8-2    8-3    8-4    8-5    8-6    8-7

9-1    9-2    9-3    9-4    9-5    9-6    9-7    9-8

10-1    10-2    10-3    10-4    10-5    10-6    10-7    10-8    10-9

## algoritmo 169

```
prog desafio34
int L,c,t;
imprima "\nABAIXO DA DIAGONAL PRINCIPAL\n";
para(L<-2;L<=10;L++)
{
    imprima "\n";
    para(c<-1;c < L;c++)
    { imprima L, "-",c," \t"; }
}
imprima "\n";
fimprog
```

### ACIMA DA DIAGONAL SECUNDÁRIA

1-1	1-2	1-3	1-4	1-5	1-6	1-7	1-8	1-9
2-1	2-2	2-3	2-4	2-5	2-6	2-7	2-8	
3-1	3-2	3-3	3-4	3-5	3-6	3-7		
4-1	4-2	4-3	4-4	4-5	4-6			
5-1	5-2	5-3	5-4	5-5				
6-1	6-2	6-3	6-4					
7-1	7-2	7-3						
8-1	8-2							
9-1								

## algoritmo 170

```
prog desafio35
int L,c,t;
imprima "\nACIMA DA DIAGONAL SECUNDARIA\n";
para(L<-1;L<=9;L++)
{
    para(c<-1;c<11 - L;c++)
    { imprima L, "-",c," \t"; }
    imprima "\n";
}
imprima "\n";
fimprog
```

## DIAGONAL SECUNDÁRIA

1-10  
2-9  
3-8  
4-7  
5-6  
6-5  
7-4  
8-3  
9-2  
10-1

### algoritmo 171

```
prog desafio36
int L,c,t;
imprima "\nDIAGONAL SECUNDARIA\n";
para(L<-1;L<=10;L++)
{
    para(t<-10 - L;t >=1;t--)
    { imprima "\t"; }
    imprima L, "-",11-L," \n";
}
imprima "\n";
fimprog
```

ABAIXO DA DIAGONAL SECUNDARIA

							2-10
							3-9      3-10
							4-8      4-9      4-10
							5-7      5-8      5-9      5-10
							6-6      6-7      6-8      6-9      6-10
							7-5      7-6      7-7      7-8      7-9      7-10
							8-4      8-5      8-6      8-7      8-8      8-9      8-10
							9-3      9-4      9-5      9-6      9-7      9-8      9-9      9-10
							10-2     10-3     10-4     10-5     10-6     10-7     10-8     10-9     10-10

## algoritmo 172

```
prog desafio37
int L,c,t;
imprima "\nABAIXO DA DIAGONAL SECUNDÁRIA\n";
imprima "\n";
para(L<-2;L<=10;L++)
{
    para(t<-11-L;t >=1;t--)
    {imprima "\t"; }
    para(c<-12-L;c<=10;c++)
    { imprima L, "-",c," \t"; }
    imprima "\n";
}
imprima "\n";
fimprog
```

E se a matriz fosse assim?

0-0	0-1	0-2	0-3	0-4	0-5	0-6	0-7	0-8	0-9
1-0	1-1	1-2	1-3	1-4	1-5	1-6	1-7	1-8	1-9
2-0	2-1	2-2	2-3	2-4	2-5	2-6	2-7	2-8	2-9
3-0	3-1	3-2	3-3	3-4	3-5	3-6	3-7	3-8	3-9
4-0	4-1	4-2	4-3	4-4	4-5	4-6	4-7	4-8	4-9
5-0	5-1	5-2	5-3	5-4	5-5	5-6	5-7	5-8	5-9
6-0	6-1	6-2	6-3	6-4	6-5	6-6	6-7	6-8	6-9
7-0	7-1	7-2	7-3	7-4	7-5	7-6	7-7	7-8	7-9
8-0	8-1	8-2	8-3	8-4	8-5	8-6	8-7	8-8	8-9
9-0	9-1	9-2	9-3	9-4	9-5	9-6	9-7	9-8	9-9

## algoritmo 173

```
prog desafio4
int L,c,t;
imprima "\nTODOS\n";
para(L<-0;L<=9;L++)
{
    para(c<-0;c<=9;c++)
    { imprima L, "-",c, " \t" ; }
    imprima "\n";
}
imprima "\n";
fimprog
```

## EXERCÍCIOS – LISTA 3

### ESTRUTURA DO PARA

#### algoritmo 174

*Imprimir todos os números de 1 até 100.*

```
prog para1
    int i;
    para( i <- 1; i <= 100; i++ )
    {   imprima i, " ";
    }
    imprima "\n";
fimprog
```

#### algoritmo 175

*Imprimir todos os números de 100 até 1.*

```
prog para2
    int i;
    para( i <- 100; i >= 1; i-- )
    {   imprima i, " ";
    }
    imprima "\n";
fimprog
```

#### algoritmo 176

*Imprimir os 100 primeiros pares.*

```
prog para3
    int i;
    para( i <- 2; i <= 200; i <- i+2 )
    {   imprima i, " ";
    }
    imprima "\n";
fimprog
```

#### algoritmo 177

*Imprimir os múltiplos de 5, no intervalo de 1 até 500.*

```
prog para4
    int i;
    para( i <- 5; i <= 500; i <- i +5 )
    {   imprima i, " ";
    }
    imprima "\n";
fimprog
```

## algoritmo 178

EET omphogia

*Imprimir o quadrado dos números de 1 até 20.*

```
prog para5
    int c;
    para( c <- 1; c <= 20; c++)
    {   imprima c ^ 2, "   ";
        imprima "\n";
    fimprog
```

## algoritmo 179

*Criar um algoritmo que imprima os números pares no intervalo de 1 a 600.*

```
prog para6
    int i;
    para( i <- 2; i <= 600; i <- i+2 )
    {   imprima i, "   ";
        imprima "\n";
    fimprog
```

## algoritmo 180

*Criar um algoritmo que imprima os números de 120 a 300.*

```
prog para7
    int c;
    para( c <- 120; c <=300; c++)
    {   imprima c , "   ";
        imprima "\n";
    fimprog
```

## algoritmo 181

*Criar um algoritmo que imprima todos os números de 1 até 100 e a soma deles.*

```
prog para8
    int i, soma;
    soma <-0;
    para( i <-1; i <= 100; i++)
    {
        soma <- soma + i;
        imprima i, "   ";
    }
    imprima "\nSomatorio de 1 a 100: ",soma;
    imprima "\n";
fimprog
```

## algoritmo 182

BVFamília

*Entrar com 10 números e imprimir a metade de cada número.*

```
prog para9
    int a;
    real numero;
    para( a <- 1; a <=10; a++)
    { imprima "\ndigite numero: ";
        leia numero;
        imprima "metade: ", numero / 2;
    }
    imprima "\n";
fimprog
```

## algoritmo 183

*Entrar com 10 números e imprimir o quadrado de cada número.*

```
prog para10
    int a;
    real numero;
    para( a <- 1; a <=10; a++)
    {
        imprima "\ndigite numero: ";
        leia numero;
        imprima "quadrado: ", numero ** 2;
    }
    imprima "\n";
fimprog
```

## algoritmo 184

*Entrar com 8 números e, para cada número, imprimir o logaritmo desse número na base 10.*

```
prog para11
    int a;
    real numero;
    para( a <- 1; a <=8; a++)
    {
        imprima "\ndigite numero: ";
        leia numero;
        se(numero > 0.)
        { imprima "\nlogaritmo: ", log(numero) /log(10.); }
        senao
        { imprima "\nNao fao logaritmo de numero negativo"; }
    }
    imprima "\n";
fimprog
```

## algoritmo 185

---

Entrar com 15 números e imprimir a raiz quadrada de cada número.

```
prog para12
  int a;
  real numero;
  para( a <= 1; a <= 15; a++)
  {
    imprima "\ndigite numero: ";
    leia numero;
    se(numero >= 0.)
    { imprima "\nraiz: ", raiz(numero); }
    senao
    { imprima "\nNao faco raiz de numero negativo"; }
  }
  imprima "\n";
fimprog
```

## algoritmo 186

---

Entrar com quatro números e imprimir o cubo e a raiz cúbica de cada número.

```
prog para13
  int a;
  real numero;
  para( a <= 1; a <= 4; a++)
  {
    imprima "\ndigite numero: ";
    leia numero;
    imprima "\ncubo: ", numero ** 3;
    se(numero < 0.)
    { imprima "\nNao faco raiz de numero negativo: ";}
    senao
    { imprima "\nraiz: ", numero **(1/3); }
  }
  imprima "\n";
fimprog
```

## algoritmo 187

---

Criar um algoritmo que calcule e imprima o valor de  $b^n$ . O valor de  $n$  deverá ser maior do que 1 e inteiro e o valor de  $b$  maior ou igual a 2 e inteiro.

```
prog para14
  int base, expo, pot, i;
  imprima "\nDigite a base inteira e maior do que 1: ";
  leia base;
  imprima "\nDigite expoente inteiro maior que 1: ";
```

```

leia expo;
se( base >= 2 && expo > 1)
{
    pot <- 1;
    para( i <- 1; i <= expo; i++)
    {
        pot <- pot * base;
    }
    imprima "\npotencia: ", pot;
}
senao
{ imprima "\nNao satisfazem";}
imprima "\n";
fimprog

```

### algoritmo 188

---

*Criar um algoritmo que imprima uma tabela de conversão de polegadas para centímetros. Deseja-se que na tabela conste valores desde 1 polegada até 20 polegadas inteiras.*

```

prog para15
    int L, c, ca;
    imprima "\nConversao de polegadas para centimetros \n";
    para(L<-1; L<=20; L++)
    { imprima "\n", L, "''' equivale(m) a ", L*2.54, " cm"; }
    imprima "\n";
fimprog

```

### algoritmo 189

---

*Criar um algoritmo que imprima a tabela de conversão de graus Celsius-Fahrenheit para o intervalo desejado pelo usuário. O algoritmo deve solicitar ao usuário o limite superior, o limite inferior do intervalo e o decremento.*

*Fórmula de conversão:  $C = 5(F - 32) / 9$*

*Exemplo:*

*valores lidos: 68 50 14*

<i>impressão:</i>	<i>Fahrenheit</i>	<i>Celsius</i>
	68	20
	50	10
	14	-10

```

prog para16
    int f1, f2, dec,t;
    real c;
    imprima "\nentre com a temperatura maior em Fahrenheit: ";
    leia f1;

```

```

imprima "\nentre com a temperatura menor em Fahrenheit: ";
leia f2 ;
imprima "\nentre com decremento: ";
leia dec ;
para( t <= f1; t >= f2; t <= t - dec)
{
    c <= 5 * (t - 32)/9;
    imprima "\ntemperatura em graus Celsius: ", c;
}
imprima "\n";
fimprog

```

### algoritmo 190

---

*Entrar com um nome, idade e sexo de 20 pessoas. Imprimir o nome se a pessoa for do sexo masculino e tiver mais de 21 anos.*

```

prog para17
int i, idade;
string nome,sexo;
para( i <= 1; i <= 10; i++ )
{
    imprima "\ndigite nome: ";
    leia nome;
    imprima "\ndigite sexo: ";
    leia sexo;
    imprima "\ndigite idade: ";
    leia idade;
    se( (strprim(sexo) == "f" || strprim(sexo)=="F") && idade >= 21)
    {
        imprima "\n", nome;
    }
    imprima "\n";
}
fimprog

```

### algoritmo 191

---

*Criar um algoritmo que leia um número que será o limite superior de um intervalo e o incremento (incr). Imprimir todos os números naturais no intervalo de 0 até esse número. Suponha que os dois números lidos são maiores do que zero. Exemplo:*

*Límite superior: 20  
Incremento: 5*

*Saída: 0 5 10 15 20*

```

prog para18
int i, limite, incr;
imprima "\nDigite o numero limite e o incremento pressionando enter apos
digitar cada um: ";
leia limite;
leia incr;

```

```
para( i <- 0; i <= limite; i <- i + incr) { imprima i, " "; }
{ imprima "\n"; }
fimprog
```

### algoritmo 192

Criar um algoritmo que leia um número que será o limite superior de um intervalo e imprimir todos os números ímpares menores do que esse número. Exemplo:

Limite superior: 15

Saída: 1 3 5 7 9 11 13

```
prog para19
int num, i, vf;
imprima "\nDigite um numero: ";
leia num;
vf <- num - 1;
para( i <- 1; i <= vf; i <- i + 2)
{ imprima i, " "; }
imprima "\n";
fimprog
```

### algoritmo 193

Criar um algoritmo que leia um número que servirá para controlar os números pares que serão impressos a partir de 2. Exemplo:

Quantos: 4

Saída: 2 4 6 8

```
prog para20
int i, num, vf;
imprima "\nDigite um numero: ";
leia num;
vf <- num * 2;
para( i <- 2; i <= vf; i <- i + 2)
{ imprima i, " "; }
imprima "\n";
fimprog
```

### algoritmo 194

Criar um algoritmo que leia um número e imprima todos os números de 1 até o número lido e o seu produto. Exemplo:

número: 3

Saída: 1 2 3

6

```
prog para21
int i, num, produto;
produto <- 1;
imprima "\nDigite um numero: ";
```

```
leia num;
para( i <= 1; i <= num ; i++)
{
    produto<- produto * i;
    imprima i, " ";
}
imprima "\nProduto de 1 a ",num,":",produto;
imprima "\n";
fimprog
```

### algoritmo 195

*Criar um algoritmo que imprima a soma dos números pares entre 25 e 200.*

```
prog para22
    int soma, i;
    soma <-0;
    para( i <- 26; i <= 198; i <- i + 2)
    {   soma <-soma + i;}
    imprima "\nSoma 26-198: ",soma;
    imprima "\n";
fimprog
```

### **algoritmo 196**

*Criar um algoritmo que leia um número (num) e imprima a soma dos números múltiplos de 5 no intervalo aberto entre 1 e num. Suponha que num será maior que zero.*

*Límite superior: 15  
( 5 10 ) – múltiplos de 5*

Saída: 15

```
prog para23
    int i, num, soma;
    soma <- 0;
    imprima "\nDigite um numero maior que zero: ";
    leia num;
    num<-num-1;
    para( i <- 5; i < num; i <- i + 5)
    {   soma <- soma + i; }
    imprima "\nSoma dos multiplos de 5: " , soma;
    imprima "\n";
fimprog
```

## algoritmo 197

*Criar um algoritmo que leia um número que servirá para controlar os primeiros números ímpares. Deverá ser impressa a soma desses números. Suponha que num será maior que zero.*

Quantos: 5

Saída: 25

( 1 3 5 7 9) – primeiros ímpares

```
prog para24
    int num,impar,i;
    imprima "\nDigite um numero maior que zero: ";
    leia num;
    impar <-0;
    para( i <- 1; i <= num * 2 ; i <- i + 2)
    {
        impar <-impar + i;
    }
    imprima "\n", impar;
    imprima "\n";
fimprog
```

### **algoritmo 198**

*Criar um algoritmo que leia os limites inferior e superior de um intervalo e imprima todos os números naturais no intervalo fechado. Suponha que os dados digitados são para um intervalo crescente. Exemplo:*

*Límite inferior: 5*

Saída: 5 6 7 8 9 10 11 12

*Límite superior: 12*

```
prog para25
    int ini, f, i;
    imprima "\nvalor inicial e final, pressionando enter apos digitar cada
um: ";
    leia ini;
    leia f;
    para( i <- ini; i <= f; i++)
    {   imprima i, "  ";
    }
    imprima "\n";
fimprog
```

algoritmo 199

*Criar um algoritmo que leia os limites inferior e superior de um intervalo e imprima todos os números múltiplos de 6 no intervalo fechado. Suponha que os dados digitados são para um intervalo crescente. Exemplo:*

*Límite inferior: 5*

Saída: 6 12

*Límite superior: 13*

```

prog para26
int ini, f, i;
imprima " digite valor inicial e valor final de um intervalo,
pressionando enter apos digitar cada um: ";
leia ini;
leia f;
se( ini % 6 == 0 )
{ ini <- ini + 6; }
senao
{ ini <- ini + (6 - (ini % 6)); }
f--;
para( i <- ini; i <= f; i <- i + 6 )
{ imprima i, " "; }
imprima "\n";
fimprog

```

## algoritmo 200

---

*Criar um algoritmo que leia os limites inferior e superior de um intervalo e o número cujos múltiplos se deseja que sejam impressos no intervalo aberto. Suponha que os dados digitados são para um intervalo crescente. Exemplo:*

Limite inferior: 3	Saída: 6 9
Limite superior: 12	
Número: 3	

```

prog para27
int ini, f, num, i;
imprima "\ndigite valor inicial, valor final de um intervalo e o numero de
que se procuram os multiplos, pressionando enter apos digitar cada um: ";
leia ini;
leia f;
leia num;
se( ini % num == 0 )
{ ini <- ini + num; }
senao
{ ini <- ini + (num - (ini % num) ); }
f--;
para( i <- ini; i <= f; i <- i + num )
{ imprima i, " "; }
imprima "\n";
fimprog

```

## algoritmo 201

---

*Criar um algoritmo que leia os limites inferior e superior de um intervalo e imprima todos os números pares no intervalo aberto e seu somatório. Suponha que os dados digitados são para um intervalo crescente. Exemplo:*

```

algoritmo Limite inferior: 3           Saída: 4 6 8 10
                                         Soma: 28
                                         Número: 3

prog para28
int ini, vf, soma, i;
imprima "\ndigite valor inicial e valor final de um intervalo,
pressionando enter apos digitar cada um: ";
leia ini;
leia vf;
soma <-0;
se( ini % 2 == 0)
{ ini <- ini + 2; }
senao
{ ini <- ini + 1; }
vf--;
para( i <- ini ; i <= vf; i <- i + 2)
{
    soma <- soma + i;
    imprima i, " ";
}
imprima "\nsoma = ", soma;
imprima "\n";
fimprog

```

## algoritmo 202

---

*Criar um algoritmo que leia um número (num) da entrada e imprima os múltiplos de 3 e 5 ao mesmo tempo no intervalo de 1 a num. Exemplo:*

Número lido: 50	Saída: 15 30 45
-----------------	-----------------

```

prog para29
int i, num;
imprima "\nEnter com um numero maior do que 15: ";
leia num;
se(num>=15)
{
    para( i <- 15 ; i <= num; i <- i + 15 )
    { imprima i, " "; }
}
senao
{ imprima "\n NAO EXISTE";}
imprima "\n";
fimprog

```

## algoritmo 203

Criar um algoritmo que leia um número (num) da entrada. Em seguida, ler n números da entrada e imprimir o triplo de cada um. Exemplo:

1º valor lido

5

5 valores lidos	valores impressos
3	9
10	30
12	36
2	6
1	3

```
prog para30
int i, num;
real num1;
imprima "\nQuantos numeros voce quer digitar? ";
leia num;
para( i <= 1 ; i <= num; i ++ )
{
    imprima "\nDigite o ", i , " numero de ", num, ": ";
    leia num1;
    imprima "\n", num1 * 3;
}
imprima "\n";
fimprog
```

## algoritmo 204

Criar um algoritmo que leia um número da entrada (num), leia n números inteiros da entrada e imprima o maior deles. Suponha que todos os números lidos serão positivos. Exemplo:

1º valor lido

3

3 valores lidos	Valor impresso
3	18
18	
12	

```
prog para31
int i, num;
real num1, maior;
imprima "\nQuantos numeros voce quer digitar? ";
leia num;
```

```

maior<-1;
para( i <= 1; i <=num; i++)
{
    imprima "\nDigite um numero: ";
    leia num1;
    se(num1 > maior)
    { maior <-num1; }
}
imprima "\nEste é o maior numero que voce digitou: ", maior;
imprima "\n";
fimprog

```

## algoritmo 205

---

*Criar um algoritmo que leia um número da entrada (num), leia n números inteiros da entrada e imprima o maior deles. Não suponha que todos os números lidos serão positivos. Exemplo:*

1º valor lido		3º valor lido	
3		Valor impresso	
	-7		-3
	-2		
	-15		

```

prog para32
int i, num;
real num1, maior;
imprima "\nQuantos numeros voce quer digitar? ";
leia num;
imprima "\nDigite um numero: ";
leia num1;
maior<- num1;
para( i <= 1; i <= num - 1; i++)
{
    imprima "Digite um numero: ";
    leia num1;
    se(num1 > maior)
    { maior <-num1; }
}
imprima "\nMaior numero digitado: ", maior;
imprima "\n";
fimprog

```

## algoritmo 206

Criar um algoritmo que leia a quantidade de números que se deseja digitar para que possa ser impresso o maior e o menor número digitados. Não suponha que todos os números lidos serão positivos.

```
prog para33
    int i, quant;
    real num, maior, menor;
    imprima "Quantos numeros voce quer digitar? ";
    leia quant;
    imprima "\nEntre com um numero: ";
    leia num;
    maior <- num;
    menor <- num;
    para( i <- 1; i < quant; i++)
    {
        imprima "\nEntre com um numero: ";
        leia num;
        se(num > maior)
        { maior <- num;}
        senao
        {
            se(num < menor)
            { menor <- num;}
        }
    }
    imprima "\nMaior: ", maior;
    imprima "\nMenor: ", menor;
    imprima "\n";
fimprog
```

## algoritmo 207

Sabendo-se que a UAL calcula o produto através de somas sucessivas, criar um algoritmo que calcule o produto de dois números inteiros lidos. Suponha que os números lidos sejam positivos e que o multiplicando seja menor do que o multiplicador.

```
prog para34
    int num1, num2, i, soma ;
    imprima "\nentre com o multiplicando: ";
    leia num1;
    imprima "\nentre com o multiplicador: ";
    leia num2 ;
    soma <-0;
    para( i <- 1; i <= num2; i++)
    { soma <- soma + num1;}
```

```
imprima "\nProduto: ", soma;
imprima "\n";
fimprog
```

605. entradas

## algoritmo 208

Criar um algoritmo que imprima os 10 primeiros termos da série de Fibonacci.

Observação: os dois primeiros termos desta série são 1 e 1 e os demais são gerados a partir da soma dos anteriores. Exemplos:

- $1 + 1 \rightarrow 2$  terceiro termo;
- $1 + 2 \rightarrow 3$  quarto termo etc.

```
prog para35
int i,ant,atual,prox;
ant<- 0;
atual<-1;
para( i <- 1; i <=10; i++)
{
    imprima "\n", atual," ";
    prox <- ant +atual;
    ant <- atual;
    atual <- prox;
}
imprima "\n";
fimprog
```

## algoritmo 209

A série de RICCI difere da série de FIBONACCI porque os dois primeiros termos são fornecidos pelo usuário. Os demais termos são gerados da mesma forma que a série de FIBONACCI. Criar um algoritmo que imprima os  $n$  primeiros termos da série de RICCI e a soma dos termos impressos, sabendo-se que para existir esta série serão necessários pelo menos três termos.

```
prog para36
int a1, a2, i, n, termo, soma;
imprima "\nEntre com 1 termo: ";
leia a1;
imprima "\nEntre com 2 termo: ";
leia a2;
imprima "\nEntre com n termos: ";
leia n;
soma <- a1 + a2;
se(n >= 3)
{
    imprima "\n", a1, " ", a2, " ";
    para( i <- 1; i <= n - 2; i++)
```

```

{
    termo <- a1 + a2;
    a1 <- a2;
    a2 <- termo;
    imprima termo, " ";
    soma <- soma + termo;
}
imprima "\nSoma: ", soma;
}
senao
{ imprima "\nnao tem serie"; }
imprima "\n";
fimprog

```

## algoritmo 210

---

A série de *FETUCCINE* é gerada da seguinte forma: os dois primeiros termos são fornecidos pelo usuário; a partir daí, os termos são gerados com a soma ou subtração dos dois termos anteriores, ou seja:

$$A_l = A_{l-1} + A_{l-2} \text{ para } l \text{ ímpar}$$

$$A_l = A_{l-1} - A_{l-2} \text{ para } l \text{ par}$$

Criar um algoritmo que imprima os 10 primeiros termos da série de *FETUCCINE*.

```

prog para37
int a1, a2, i, termo;
imprima "\nEntre com 1 termo: ";
leia a1;
imprima "\nEntre com 2 termo: ";
leia a2;
imprima "\n", a1, " ", a2, " ";
para( i <- 3; i <= 10; i++)
{
    se( i % 2 == 0)
    { termo <- a2 - a1; }
    senao
    { termo <- a2 + a1; }
    imprima termo, " ";
    a1 <- a2;
    a2 <- termo;
}
imprima "\n";
fimprog

```

## algoritmo 211

---

Criar um algoritmo que imprima todos os números inteiros e positivos no intervalo aberto entre 10 e 100 de modo que: