CAPÍTULO 1

Algoritmos, Tipos de dados, Variáveis, Operadores e Expressões

CAPÍTULO 1

LÓGICA

O que é lógica?
É a arte de pensar corretamente. A lógica ensina a colocar ordem no pensamento. Uma de suas preocupações é determinar quais operações são válidas e quais não são. E quais operações antecedem outras.

Exemplos:

1) Uma pessoa adulta e consciente, para tomar banho, primeiro tira a roupa usando a lógica de estabelecer um contato direto entre a sua pele e a água e também para não molhar a sua roupa. (Lógica de ordem)

2) Moramos em três pessoas.
Nenhum de nós dois quebrou o vaso de porcelana.
Quem quebrou o vaso? (Lógica de exclusão)

3) Gerson é cientista.
Todo cientista é estudioso.
Logo, Gerson é estudioso. (Lógica dedutiva)

Exercício 1.1) Um homem precisa atravessar um rio com um barco que possui capacidade de carregar apenas ele mesmo e mais uma de suas três cargas, que são: um lobo, um bode e um maço de alfafa. O que o homem deve fazer para conseguir atravessar o rio sem perder suas cargas? Qual é a lógica para uma travessia segura?

LÓGICA DE PROGRAMAÇÃO

Lógica de Programação é a técnica de desenvolver sequências lógicas para atingir um determinado objetivo. Essas sequências lógicas são adaptadas para uma linguagem de computador, pelo programador, a fim de produzir *software.*

Uma sequência lógica é denominada algoritmo. Como podemos ver a lógica de programação trata basicamente de construir algoritmos que serão transformados em programas de computador.

ALGORITMO

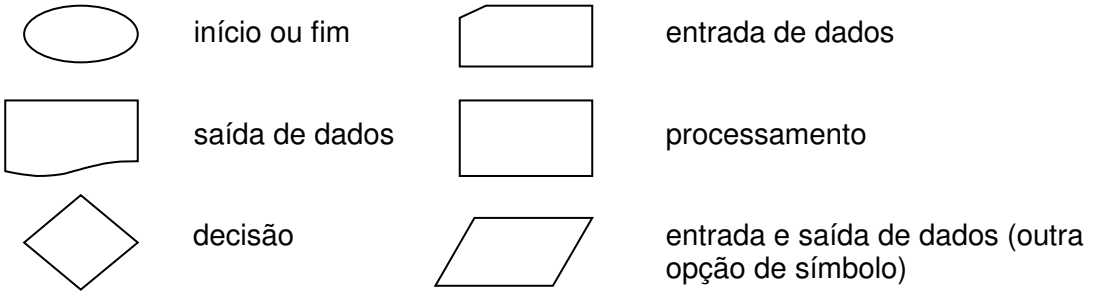
É uma sequência lógica de passos que visa atingir um objetivo bem definido.
Exemplos: uma receita de bolo
a troca de um pneu furado
o cálculo da média de um aluno
As formas mais conhecidas de representar um algoritmo são:

Descrição narrativa: expressa em linguagem natural

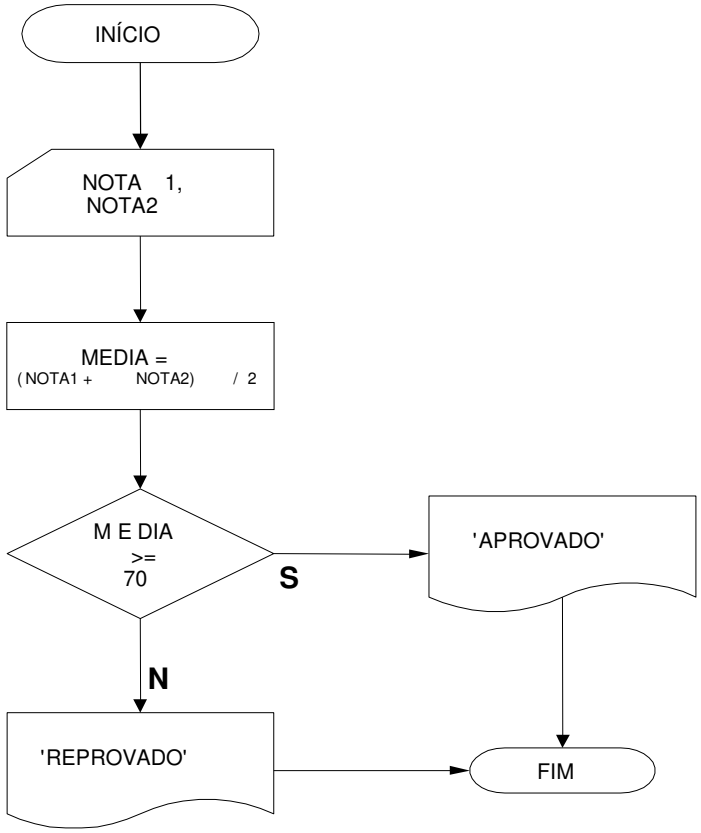
Exemplo - A troca de um pneu furado:

1) afrouxar ligeiramente as porcas
2) suspender o carro
3) retirar as porcas e o pneu
4) colocar o pneu reserva
5) apertar as porcas
6) abaixar o carro
7) dar o aperto final nas porcas

Fluxograma convencional: expresso com desenhos (símbolos)



Exemplo: Cálculo da média de um aluno e sua situação de aprovação



Exercício 1.2) Faça um algoritmo em fluxograma para calcular e mostrar o perímetro de um triângulo dados seus lados. E também verificar se este perímetro é superior a 1metro.

Pseudocódigo: assemelha-se bastante à forma com que os programas são escritos. Também chamado de PORTUGOL (português + algol).

```
Algoritmo MEDIA_FINAL
declare
        NOTA1, NOTA2, MEDIA numérico; // primeira e segunda notas e média
início
        escreva "Entre com as 2 notas";
        leia NOTA1, NOTA2;                  // entrada das 2 notas
        MEDIA ← (NOTA1 + NOTA2) / 2;    // cálculo da média
        se (MEDIA >= 70)                    // análise da aprovação
            então escreva "Aprovado";
            senão escreva "Reprovado";
fim.
```

Escrever um algoritmo, em descrição narrativa, para trocar uma lâmpada queimada no teto:

- PEGUE UMA ESCADA
- POSICIONE-A EMBAIXO DA LÂMPADA
- BUSQUE UMA LÂMPADA NOVA
- SUBA NA ESCADA
- RETIRE A LÂMPADA VELHA
- COLOQUE A LÂMPADA NOVA

A **sequenciação** é uma convenção com o objetivo de reger o fluxo de execução, determinando qual ação vem a seguir.

Está correto o algoritmo? – Acho que sim.
E se a lâmpada não estiver queimada?
O algoritmo faz com que ela seja trocada do mesmo modo. Ele não prevê esta situação.
- Realmente.....
O que temos que incluir no algoritmo? - Vamos incluir uma condição (verificação)

- PEGUE UMA ESCADA
- POSICIONE-A EMBAIXO DA LÂMPADA
- BUSQUE UMA LÂMPADA NOVA
- **LIGUE O INTERRUPTOR. SE A LÂMPADA NÃO ACENDER, ENTÃO:**
  - SUBA NA ESCADA
  - RETIRE A LÂMPADA VELHA
  - COLOQUE A LÂMPADA NOVA

O que ocorreu foi a inclusão de um **teste seletivo** (condição) que determina qual ou quais ações serão executadas dependendo do resultado da inspeção da condição.

O algoritmo pode ser otimizado? – Sim.
O que poderíamos fazer? – Colocar a verificação no início.

- **LIGUE O INTERRUPTOR. SE A LÂMPADA NÃO ACENDER, ENTÃO:**
  - PEGUE UMA ESCADA
  - POSICIONE-A EMBAIXO DA LÂMPADA
  - BUSQUE UMA LÂMPADA NOVA
  - SUBA NA ESCADA
  - RETIRE A LÂMPADA VELHA
  - COLOQUE A LÂMPADA NOVA

A solução parece adequada. Mas e se a lâmpada nova não funcionar?

- LIGUE O INTERRUPTOR. SE A LÂMPADA NÃO ACENDER, ENTÃO:
  - PEGUE UMA ESCADA
  - POSICIONE-A EMBAIXO DA LÂMPADA
  - BUSQUE UMA **CAIXA DE LÂMPADAS**

- SUBA NA ESCADA
- RETIRE A LÂMPADA VELHA
- COLOQUE A LÂMPADA NOVA
- **SE A LÂMPADA NÃO ACENDER, ENTÃO:**
  - **RETIRE A LÂMPADA E COLOQUE OUTRA**
- **SE A LÂMPADA NÃO ACENDER, ENTÃO:**
  - **RETIRE A LÂMPADA E COLOQUE OUTRA**
- **SE..**
- **.....**

Até quando???

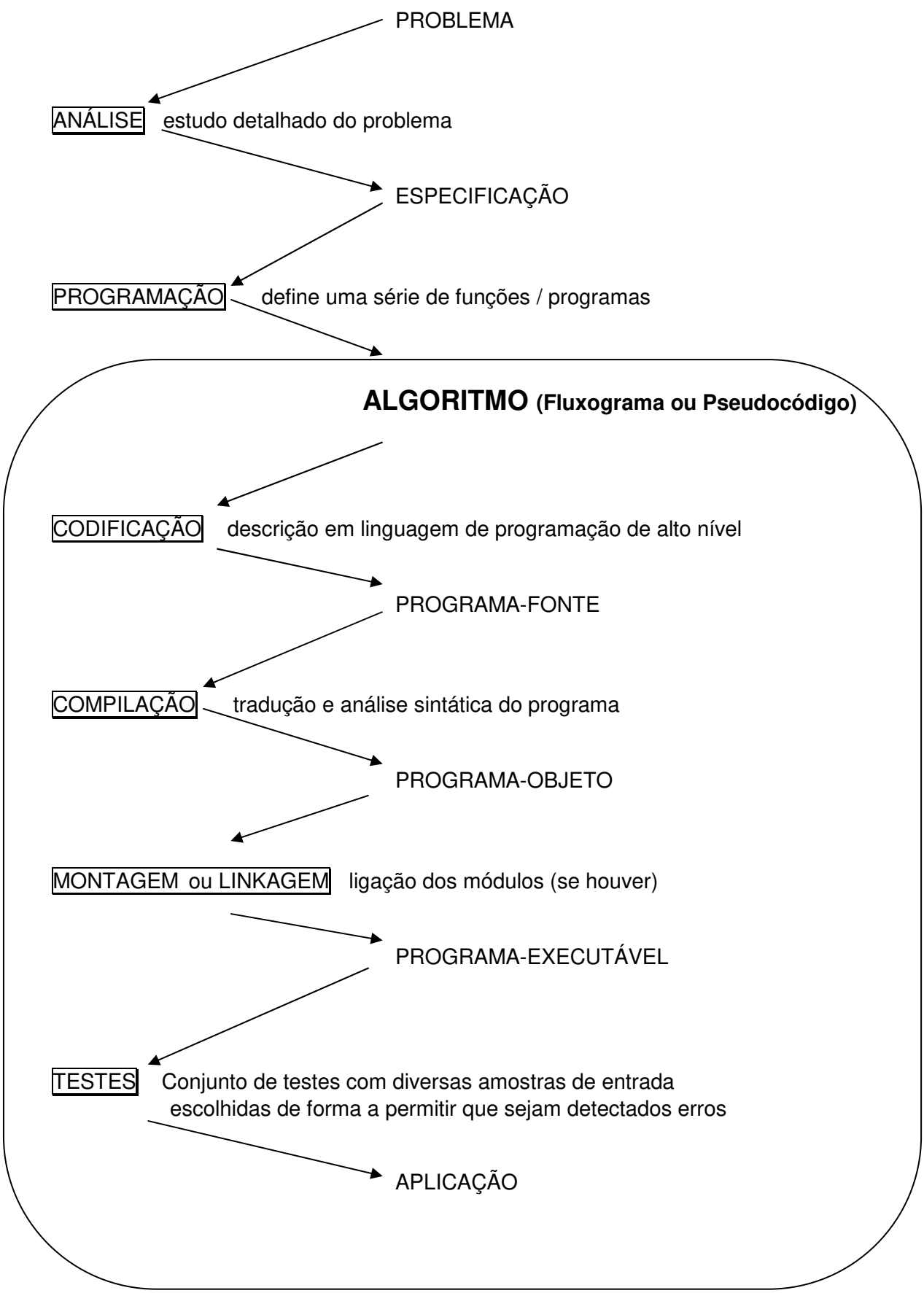
- LIGUE O INTERRUPTOR. SE A LÂMPADA NÃO ACENDER, ENTÃO:
  - PEGUE UMA ESCADA
  - POSICIONE-A EMBAIXO DA LÂMPADA
  - BUSQUE UMA CAIXA DE LÂMPADAS
  - SUBA NA ESCADA
  - RETIRE A LÂMPADA VELHA
  - COLOQUE A LÂMPADA NOVA
  - **ENQUANTO A LÂMPADA NÃO ACENDER, FAÇA**
    - **RETIRE A LÂMPADA**
    - **COLOQUE OUTRA LÂMPADA**
  - CONTABILIZA UM SOQUETE TROCADO

O algoritmo repete os dois últimos comandos até que a lâmpada acenda, ou seja, estabelece um **fluxo repetitivo** de comandos. Percebe-se que o número de repetições é indefinido, e que depende apenas da condição estabelecida, o que leva a repetir as ações até se alcançar o objetivo: trocar a Lâmpada queimada por outra que funcione.

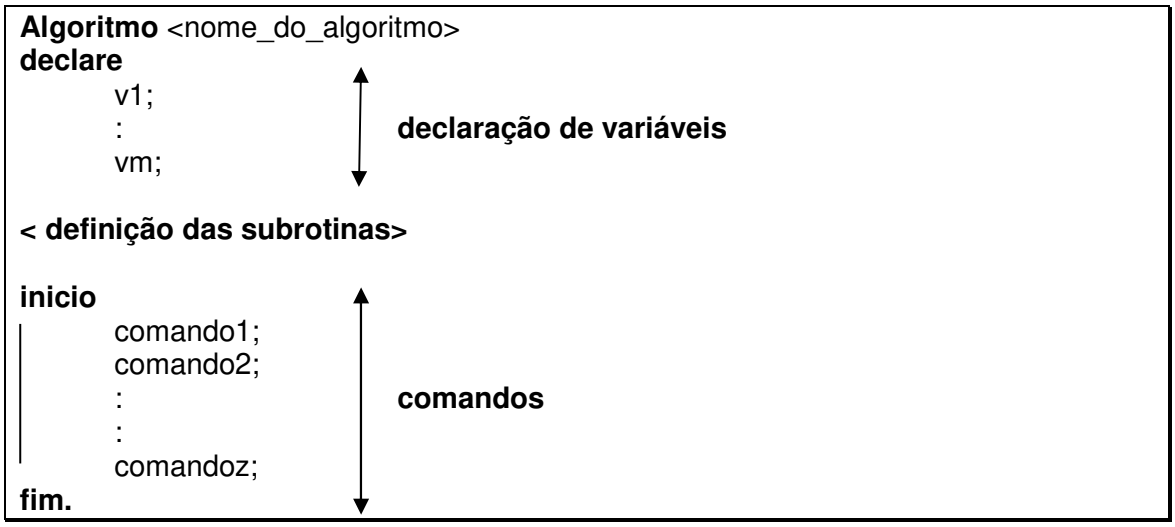
E se tivéssemos 10 lâmpadas para trocar em lugares distintos?
Teríamos que ter um algoritmo para cada lâmpada? Não

- **ENQUANTO A QUANTIDADE DE SOQUETES TROCADOS FOR MENOR QUE 10, FAÇA:**
  - LIGUE O INTERRUPTOR. SE A LÂMPADA NÃO ACENDER, ENTÃO:
    - PEGUE UMA ESCADA
    - POSICIONE-A EMBAIXO DA LÂMPADA
    - BUSQUE UMA CAIXA DE LÂMPADAS
    - SUBA NA ESCADA
    - RETIRE A LÂMPADA VELHA
    - COLOQUE A LÂMPADA NOVA
    - ENQUANTO A LÂMPADA NÃO ACENDER, FAÇA
      - RETIRE A LÂMPADA
      - COLOQUE OUTRA LÂMPADA
  - MAIS UM SOQUETE TROCADO

PROCESSO DE AUTOMAÇÃO DE UM PROBLEMA



ESTRUTURA DE UM ALGORITMO EM PSEUDOCÓDIGO



Onde:

Algoritmo é uma palavra que indica o início de um algoritmo em forma de pseudocódigo

<nome_do_algoritmo> é um nome dado ao algoritmo que o distingue dos demais

declare indica o início da declaração das variáveis

declaração das variáveis é o local onde são declaradas as variáveis globais usadas no algoritmo principal e, eventualmente nas subrotinas

definição das subrotinas é o local onde são definidas as subrotinas, se houver (matéria abordada no Capítulo 6)

inicio e **fim** são as palavras que delimitam o conjunto de instruções do corpo do algoritmo

Observações:

Algoritmo, **declare**, **inicio** e **fim** são consideradas palavras reservadas da

pseudo-linguagem e, portanto, não se pode declarar variáveis com estes nomes.

Existe um conjunto de regras e convenções para o desenvolvimento de algoritmos:

tipos de dados e variáveis
operadores e expressões
comando de atribuição
comandos de entrada e saída
estrutura sequencial
estruturas condicionais
estruturas repetitivas
variáveis indexadas homogêneas
sub-rotinas, funções
variáveis indexadas heterogêneas
arquivos

TIPOS DE DADOS

Durante a execução de qualquer programa/aplicação, os computadores estão manipulando informações representadas pelos diferentes tipos de dados. Para ter acesso a estas informações, o computador precisa guardá-las em sua memória, e isto é feito com o uso de **variáveis**. Todos os computadores trabalham com sistema numérico binário (0 zeros e 1 uns). Cada dígito binário (0 ou 1) ocupa uma porção de memória chamada *bit*, e um conjunto de 8 *bits* é denominado *byte*. As entidades, variáveis, são nomes de locais onde se pode colocar qualquer valor do conjunto de valores disponíveis abaixo:

Dado numérico inteiro: é aquele pertencente ao conjunto dos números inteiros (Z). Pode ser positivo ou negativo e não possui parte fracionária. Embora possa ser representado pela classe dos números reais, é classificado como dado do tipo inteiro por permitir economia de espaço de memória e maior velocidade de cálculos. Em geral, o dado inteiro ocupa 2 bytes [de -32768 a +32767].

Exemplos: 224 0 -112

Dado numérico real: é aquele que pode possuir componentes decimais ou fracionários. Pode ser negativo ou positivo. Os dados reais seguem a notação da língua inglesa, ou seja, a parte decimal é separada da parte inteira por um . (ponto) e não por uma , (vírgula). Em geral, o dado real ocupa 4 bytes.

Exemplos: 224.01 -13.3 0.0 144.0

Os dados literais (caracteres) são utilizados para armazenar nomes, endereços, mensagens, etc. Os símbolos permitidos para estes tipos de dados são letras maiúsculas ou minúsculas, números (não podem ser utilizados para cálculos), símbolos especiais (&, #, @, ?, +) e o espaço em branco. Esses tipos de dados são também conhecidos como alfanuméricos ou cadeia de caracteres.

Dado literal char: utilizado para armazenar um único caracter. O dado do tipo char ocupa 1 byte e é representado entre apóstrofes ‘ ’.

Exemplos: ‘u’ ‘5’ ‘C’ ‘%’ ‘ ’

Dado literal string ou char[xx]: consiste de uma sequência de caracteres. Este tipo de dado, quando armazenado na memória do computador, ocupa 1 byte para cada caractere. É representado entre aspas “ ”.

Exemplos: “qual ?” “1234.56” “alunos@inatel.br” “1 + 2”

Dado lógico ou *booleano*: usado para representar dois únicos valores lógicos possíveis: verdadeiro *(true)* e falso *(false)*. O dado do tipo lógico ocupa 1 byte. É chamado de *booleano* devido à contribuição do filósofo e matemático inglês George Boole da área de lógica matemática.

Exemplos: **true** (**verdadeiro)** **false (falso)**

COMENTÁRIOS

Os comentários são usados para aumentar a clareza de um programa. São utilizados para documentar os códigos. O trabalho de manutenção e modificação dos mesmos fica facilitado se os comentários forem bem colocados.

Ao colocarmos comentários em um programa devemos sempre procurar nos colocar no lugar de uma pessoa que não participou da elaboração do mesmo, e imaginar as possíveis dúvidas que esta pessoa poderia ter ao tentar entender o programa.

Nos programas iremos usar as seguintes formas de comentários para facilitar a compreensão da solução apresentada:

/* entre parênteses e asteriscos */

// após barras duplas até o final da linha

Os comentários não são analisados pelo compilador.
Fique atento também para o excesso de comentários, pois é tão prejudicial quanto a ausência deles.

SINTAXE / SEMÂNTICA

Antes de iniciar a descrição do conjunto de regras e convenções que serão usadas no desenvolvimento de programas, vamos definir alguns termos:

sintaxe é a forma como os comandos dever ser escritos, a fim de que possam ser entendidos pelo compilador (tradutor)

semântica é o significado do comando, ou seja, o conjunto de ações que serão executadas pelo computador durante a execução do comando.

A partir do próximo capítulo, todos os comandos serão apresentados por meio de sua sintaxe e sua semântica, ou seja, a forma como devem ser escritos e as ações que cada um executa.

VARIÁVEIS

A variável é uma entidade destinada a guardar uma informação. Cada variável corresponde a uma posição de memória cujo conteúdo pode variar ao longo do tempo, durante a execução de um programa. Mas, em um determinado instante, a posição de memória pode ter somente um valor. Durante a execução do programa o valor pode ser alterado, porém o valor antigo será perdido.

Toda variável é identificada por um nome ou **identificador**. Um identificador é formado por um ou mais caracteres. O primeiro deve ser, uma letra (maiúscula ou minúscula) ou o caractere sublinhado (_) e os demais, letras (maiúsculas ou minúsculas) e/ou dígitos. Não é permitido o uso de símbolos especiais, exceto o sublinhado (_) e o uso de palavras reservadas (palavras-chaves que pertencem à linguagem de programação).

Identificadores permitidos	Identificadores não permitidos
a	5b
nota	e(13)
a32b	a:b
_ndepend	123
x25	for
matricula	nome comprido
nome_comprido	matrícula
caixa_preta	if

Os seguintes nomes são distintos: PESO Peso peso peSo

As linguagens de programação, na maioria, são *case sensitive*.

Deve-se criar locais na memória com o nome da variável (identificador) e o seu tipo. O tipo de uma variável informa a quantidade de memória, em bytes, que ela ocupará e o modo como o valor deverá ser armazenado e interpretado. Veja a sintaxe de declaração!

SINTAXE DE VARIÁVEIS EM ALGORITMOS

Todas as variáveis utilizadas em algoritmo devem ser definidas antes de serem utilizadas, pois o compilador precisa reservar um espaço de memória para cada uma delas. Utiliza-se a palavra-chave **declare**, o(s) nome(s) da(s) variável(eis) e o tipo de dado da(s) variável(eis).

```
declare
        nome1  tipo1; // comentários.....
        nome2, nome3 tipo2;
        /* comentários.....
           .....*/
        nomem tipom;
```

Onde:

- nome1, nome2, nomem são os nomes das variáveis a serem usadas no algoritmo e devem ter no máximo 247 caracteres
- variáveis do mesmo tipo, devem ser declaradas na mesma linha, separadas por vírgula (,)
- variáveis de tipos diferentes devem ser declaradas em linhas diferentes
- a declaração de tipos distintos é separada por ponto-e-vírgula (;)
- é recomendável que os nomes das variáveis sejam os mais significativos possíveis, ou seja, que o nome reflita o que está armazenado nela. Isto ajuda no entendimento do algoritmo
- tipo1, tipo2, tipom são os tipos de dados das variáveis (numérico, literal, e lógico)

Exemplo:

```
declare
        media_aluno, matricula, idade numérico;
        sexo, nomedoaluno literal;
        teste lógico;
```

SINTAXE DE VARIÁVEIS EM PROGRAMAS EM C++

Todas as variáveis utilizadas no programa devem ser definidas antes de serem utilizadas, pois o compilador precisa reservar um espaço de memória para cada uma delas.

Tipos básicos de variáveis em C++: *char, int, float, double, string, bool, char[xx].*

Alguns tipos de dados podem ser acompanhados por um *modificador (short, long e unsigned).*

Tipo	Descrição	Bytes	Exemplos
int	números inteiros	2	10; 0; -454; 230
long int	números inteiros (limite maior)	4	10; 0; -454; 230
float	números reais	4	2.1; -82.34; 0; 19.8
double	números reais (limite maior)	8	2.1; -82.34; 0; 19.8
long double	números reais (limite ainda maior)	10	2.1; -82.34; 0; 19.8
char	um caracter	1	'a'
string	literal comprido (sequência de vários caracteres)	∞	"Instituto Nacional"
char [10]	literal comprido (sequência de 9 caracteres)	9	"Instituto"

Algumas linguagens de programação dispensam a declaração de variáveis. O espaço de memória é reservado à medida que elas são encontradas no programa. Exemplo: BASIC, PYTHON, RUBY.

Sintaxe:

```
tipo1 nome1; // comentários.....
tipo2 nome2, nome3;
/* comentários.....
   .....*/
tipom nomem = valorinicial;
```

Onde:

- nome1, nome2, nomem são os nomes das variáveis a serem usadas no programa e devem ter no máximo 247 caracteres
- variáveis do mesmo tipo, devem ser declaradas na mesma linha, separadas por vírgula (,)
- variáveis de tipos diferentes devem ser declaradas em linhas diferentes
- a declaração de tipos distintos é separada por ponto-e-vírgula (;)
- é recomendável que os nomes das variáveis sejam os mais significativos possíveis, ou seja, que o nome reflita o que está armazenado nela. Isto ajuda no entendimento do programa
- tipo1, tipo2, tipom são os tipos de dados das variáveis (*char, int, float, double, string, bool, char[xx])*
- se desejar dar um valor inicial para a variável, iguale-a (=) ao valor desejado

Todo compilador gera uma tabela de mapeamento/alocação que contém o nome da variável, seu tipo (para saber quantos *bytes* ocupará) e seu endereço binário inicial de armazenamento. A primeira posição deste conjunto de *bytes* (variável) é arbitrária e sua escolha, geralmente, é feita automaticamente pelo compilador.

Desta maneira, quando queremos buscar algum dado na memória, basta utilizarmos o nome da variável, que o computador, por meio da tabela de mapeamento, busca o conteúdo automaticamente.

Exemplo:

float media_aluno;
int matricula, idade;
char sexo;
string nomedoaluno;
bool teste;

Exercício 1.3) Verifique quais os identificadores são válidos. Se não, explique por quê:

a) abc#3
b) _matric
c) numero_pontos_obtidos
d) acd1
e) aluno@inatel.br
f) 1nota
g) x y za
h) sal/hora

Exercício 1.4) Supondo que as variáveis **nt, na, nmat** e **sx** sejam utilizadas para armazenar a nota do aluno, o nome do aluno, o número de matrícula e o sexo, respectivamente, declare-as corretamente, associando os tipos adequados aos dados que serão armazenados.

Exercício 1.5) Encontre os erros das seguintes declarações de variáveis:

int endereço-aluno, nfilhos, 1peso;
bool Lâmpada:;
string idade;

EXPRESSÕES ARITMÉTICAS, LÓGICAS E LITERAIS

Expressão é uma combinação de variáveis e operadores, que, uma vez resolvida/avaliada, resulta em um valor.

EXPRESSÕES ARITMÉTICAS

Denomina-se expressão aritmética aquela cujos operadores são aritméticos e cujos operandos são variáveis do tipo numérico.

Exemplos de expressões aritméticas onde aparecem as operações de adição, subtração, multiplicação, divisão real, resto de divisão, potenciação e radiciação:

a) x + y
b) x – y
c) 2 * nota
d) a / b
e) a % b
f) **sqrt**(p)
g) **pow**(soma,2)
h) **pow**(soma, z)
i) **pow**(a, (1/2))
j) x++
k) x--

Irá ocorrer um erro de execução/compilação quando:
b for zero em uma expressão da forma total/b e em
a % b quando a e b não forem variáveis inteiras

A notação utilizada para expressões aritméticas nos programas é, basicamente, a mesma da Matemática, a menos das seguintes restrições:

a) não é permitido omitir o operador de multiplicação (*). Isto evita confusão, pois na expressão ab + c, como saber se ab é o nome de uma variável ou a multiplicação entre os conteúdos de duas variáveis, cujos nomes são a e b?

b) potenciação e radiciação: **pow**(_,_)

c) nas expressões aritméticas, as operações guardam entre si uma relação de prioridade.

PRIORIDADE DAS OPERAÇÕES	
PRIORIDADE	**OPERAÇÃO**
1a.	POTENCIAÇÃO, RADICIAÇÃO
2a.	MULTIPLICAÇÃO, DIVISÃO, RESTO DA DIVISÃO
3a.	ADIÇÃO, SUBTRAÇÃO

Para se obter uma sequência de cálculo diferente, vários parênteses podem ser usados para quebrar as prioridades definidas. Não é permitido o uso de colchetes e chaves. Estes símbolos são utilizados nos programas para outras finalidades (por exemplo, colchetes em variáveis indexadas).
Exemplo: A - B * (C + D / (E - 1) - F) + G

Se na matemática temos as expressões (a e b):

a) {43 . [55 : (30 + 2)]} , na forma computacional tem-se:

(43 * (55 / (30 + 2)))

b) $\frac{2+a}{b-3} - 2x + x^3$, na forma computacional tem-se:

(2+a)/(b-3) – 2*x + **pow**(x,3)

Além das operações básicas, podem-se usar nas expressões aritméticas algumas funções comuns na Matemática:

sin(a) – seno de a
cos(a) – cosseno de a
log10(a) - logaritmo na base 10 de a
log(a) - logaritmo natural de a
exp(a) – a-ésima potência de *e* (número de Euler = 2.718281...)
abs(a) - valor absoluto de a
ceil(a) – arredonda um número real para cima (inteiro superior)
floor(a) – arredonda um número real para baixo (inteiro inferior)
random(a) – número decimal aleatório entre 0 e a inteiro
e outras.
Exemplos:

x + **sin**(a+b+c)
(nota % 2) * 100 + t
x + **log**(y) - **abs**(a-b)

Exercício 1.6) Sendo a, b, x, y, variáveis do tipo numérico, fornecer os resultados para cada uma das seguintes expressões, onde a = 10, b = 3, x = 2.5 e y = 1.2

a) a / b , a % b

b) x / 2 , x % 2

c) **floor**(**pow**(b,2) + x), **ceil**(a/3 + 1) , **floor**(y – x)

d) **abs**(a – **pow**(b,3)), **floor**(x-3.2), **abs**(a - b), **ceil**(a - x)

e) (b+y) / (x + 1)

EXPRESSÕES LÓGICAS

Denomina-se expressão lógica a expressão cujos operadores são lógicos e cujos operandos são relações e variáveis de diversos tipos.

OPERADORES RELACIONAIS - indicam a comparação a ser realizada entre os termos da relação:

== igual a	< menor que
!= diferente de	>= maior ou igual a
> maior que	<= menor ou igual a

O resultado obtido de uma relação é sempre um valor lógico: falso ou verdadeiro. Exemplos: a != b; nome == "JOAO"; x == 1; (*pow*(b,2) - 4*a*c) < 0

Exercício 1.7) Dadas as variáveis numéricas x, y, z e as variáveis **strings** nome e cor, observar os resultados obtidos para as relações a partir dos valores atribuídos a estas variáveis.

	VARIÁVÉIS					**RELAÇÕES**		
	x	**y**	**z**	**cor**	**nome**	**(pow(x,2)+y) > z**	**cor == "AZUL"**	**nome!= "JOSÉ"**
a)	1	2	5	"AZUL"	"PAULO"			
b)	1	2	1	"BRANCO"	"JOSÉ"			

OPERADORES LÓGICOS - e (&&)- para a CONJUNÇÃO
ou (||) - para a DISJUNÇÃO
não (!) - para a NEGAÇÃO

TABELAS-VERDADE

É o conjunto de todas as possibilidades combinatórias entre os diversos valores lógicos, os quais se encontram em apenas duas situações (F ou V), e um conjunto de operadores lógicos.

A	***B***	***A && B***
F	F	F
F	V	F
V	F	F
V	V	V

A	***!A***
F	V
V	F

A	***B***	***A || B***
F	F	F
F	V	V
V	F	V
V	V	V

PRIORIDADE

Como podemos ter mais de um operador lógico na mesma expressão, a ordem em que são efetuadas estas operações afeta o resultado final. Assim, como acontece

entre as operações aritméticas, também existe uma relação de prioridade entre os operadores lógicos conforme tabela abaixo.

PRIORIDADE	***OPERADOR***
1ª.	ARITMÉTICO
2ª.	RELACIONAL
3ª.	NÃO
4ª.	E
5ª.	OU

Exercício 1.8) Dadas as variáveis numéricas x, y e z, contendo os valores 2, 5 e 9, respectivamente; a variável **string** nome, contendo "MARIA"; e a variável **bool** sim, contendo o valor lógico *false,* observar os resultados obtidos das expressões lógicas a seguir:

a) (x + y > z) && (nome == "MARIA")

b) (nome == "JORGE") && !sim || (**pow**(x,2) < z+10)

EXPRESSÕES LITERAIS

Uma expressão literal é aquela formada por operadores literais e operandos que são variáveis do tipo **string, char[xx]** e **char**.
As operações entre valores literais são bastantes diversificadas e dependem das características de cada Linguagem de Programação.

Exemplos:

CONCATENA (char1, char2)
concatena a cadeia de caracteres identificada por char2 à cadeia char1

LOCALIZA (char1, char)
retorna a posição na cadeia char1, onde o caracter char é encontrado pela primeira vez

LEN (char1)
obtém o tamanho de uma cadeia de caracteres

Além da concatenação, da localização e da obtenção do tamanho de *strings*, é comum a existência de outras operações desta natureza nas Linguagens de Programação, sendo que normalmente elas são encontradas na forma de funções: n primeiros caracteres de um *string*, n últimos caracteres de um *string*, deleção de caracteres de um *string*, inserção de caracteres em um *string*, etc.

Exercícios propostos do Capítulo 1:

P1.1) “Eu nunca me importo em usar cinto de segurança quando tenho que ir ao supermercado local, porque eu não estarei dirigindo a mais de 40 Km/h”.

O autor da frase aparentemente assume que acidentes de trânsito:

a) somente ocorrem quando o motorista não está em sua vizinhança.
b) nunca ocorrem quando o carro está a menos de 40 km/h.
c) nunca ocorrem quando o carro está andando exatamente a 40 km/h.
d) sempre ocorrem quando o carro está exatamente a 40 km/h.
e) somente ocorrem quanto o carro está a mais de 40 km/h.

P1.2) Cinco times ibéricos - Antares, Bilbao, Cascais, Deli e Elite – disputam um campeonato de basquete e, no momento, ocupam as cinco primeiras posições na classificação geral. Sabe-se que:

- Antares está em primeiro lugar e Bilbao está em quinto;
- Cascais está na posição intermediária entre Antares e Bilbao;
- Deli está à frente do Bilbao, enquanto que o Elite está imediatamente atrás do Cascais.

Nestas condições, quem está em segundo lugar?

P1.3) Três macacos sábios têm os seguintes nomes: João, Dito e Luís. Seus sobrenomes são Ferpa, Salto e Melão, não necessariamente nessa ordem. Um deles não vê, outro não fala e outro não ouve, também não necessariamente nessa ordem. Dito lamenta que seu amigo Ferpa não possa ouvir. Luís e Salto adoram ver as macaquices mútuas. Aquele que não ouve vive assistindo às provocações entre João e Melão. Qual é o nome completo (nome e sobrenome) e a característica de cada um (cego, surdo ou mudo)?

P1.4) Faça um algoritmo em fluxograma para mostrar o resultado da divisão de dois números lidos. Verifique a possibilidade de não haver divisão (divisão por zero).

P1.5) Para que utilizamos uma variável em programação?

P1.6) Verifique quais nomes de variáveis são válidos. Se não forem, explique por quê.

a) média
b) xyz#3
c) 1nota
d) num_pontos
e) p&aa
f) _123

P1.7) Declare variáveis para as seguintes informações:

número da carteira de identidade
resistência de um resistor
sexo do funcionário
número de dependentes

P1.8) Encontre os erros nas declarações das variáveis:

string salario, nome;
bool num-depend;
int ende-func, c_custo;

P1.9) Avalie as expressões e indique o seu resultado:

a) 100 / 3 % 3

b) (4 / 2 + 4.2 >= 4.2) **&&** *true*

c) **abs**(2.1) + **floor**(4.2)

P1.10) Substitua as expressões matemáticas por expressões algorítmicas:

a) $-b+\sqrt{b^2-4ac}$

b) $\alpha\, e^{-\alpha x}$

c) $\sqrt{\frac{2}{m}}\,\mathrm{sen}\left[\frac{\pi}{4m}(2n+1)\right]$

P1.11) Seja a seguinte declaração de variáveis:

int x, a, b;
float y;
string s;
bool teste;

Se os valores destas variáveis são x = 1, y = 4.2, a = 2, b = 4, s = “Zé” e teste = *true*, avalie as expressões abaixo indicando o tipo de dado do resultado e o valor resultante.

a) a + b / 2 + 7
b) s == “Maria” && **floor**(y/a) == 2
c) b / a + y < 4.2 || teste
d) **random**(b) > 10
e) **abs**(-a) – **floor**(x) + **ceil**(y)

P1.12) Escreva uma expressão lógica que resulte em **verdadeiro** se um dado número é inteiro, e, retorne, **falso**, se não for.

P1.13) Escreva uma expressão lógica que resulte em **verdadeiro** se um dado número inteiro é ímpar e múltiplo de 3, e, retorne **falso**, se não for.

P1.14) Escreva uma expressão lógica que resulte **verdadeiro** se o ano for bissexto e **falso** se não for. Um ano é bissexto se for divisível por 4, mas não for divisível por 100. Um ano também é bissexto se for divisível por 400.

P1.15) Substitua as expressões matemáticas por expressões programáveis:

a) $\sqrt{\left[\left(|x| + \lambda\right).y\right]^{w}}$

b) $\frac{sen(x)}{\cos(x)} + \sqrt[3]{|b|} + e^{\frac{\pi}{2x}}$

P1.16) Marque (**F**)alsa para a sentença falsa ou (**V**)erdadeira para a sentença verdadeira. Quando **for falsa**, **justifique o porquê** ou **escreva o correto**:

a) ( ) O compilador traduz o programa e detecta erros de sintaxe e de lógica.

b) ( ) Sendo b, x e y variáveis do tipo numérico; b = 3, x = 2.2 e y = 1.5. O resultado fornecido por **floor(b+y) % ceil(x+1)** é 0.

c) ( ) Sejam a e b variáveis do tipo numérico, a = 10 e b = 2. O resultado fornecido por **abs(a – pow(b,3))** é 1.

d) ( ) Os identificadores nome_usuario, zfx#2, _pontos e p_origem2 são nomes válidos para variáveis.

e) ( ) Programa é uma sequência lógica de passos, escrita em linguagem de alto nível, que visa atingir um objetivo bem definido.

f) ( ) O caracter é uma letra, um dígito, um espaço em branco ou um símbolo especial.

g) ( ) Com a evolução da tecnologia, os computadores digitais passaram a trabalhar com letras ao invés de *bits*.

h) ( ) Dadas as variáveis y e z contendo os valores 9 e 5, respectivamente, e a variável lógica tem, contendo o valor **false**, o resultado obtido de (**!**(tem) **&&** (z % y) == 4) é ***true***.

i) ( ) Um caracter é representado por um bit.