Introdução à Lógica

Introdução à Lógica

- **Objetivo:**
- Abordar o conceito de lógica como ciência;
- Destacar o uso da lógica de maneira muitas vezes incondicional, nas tarefas do dia-a-dia;
- Usar o raciocínio lógico para a tomada de decisões e para a resolução de problemas.

Introdução à Lógica

- Aristóteles, o criador da lógica;
- Lógica e razão;
- A palavra lógica é originária do grego *logos,* que significa linguagem racional;
- “Lógica é a análise das formas e leis do pensamento.” *(Michaelis)*
- Não se preocupa com a produção do pensamento, mas sim com a maneira pela qual um pensamento ou idéia é organizado e apresentado.

Lógica - Argumentos

É por meio do encadeamento dos argumentos de uma idéia ou pensamento que se chega a uma conclusão;

Os argumentos podem ser: dedutivos ou indutivos:

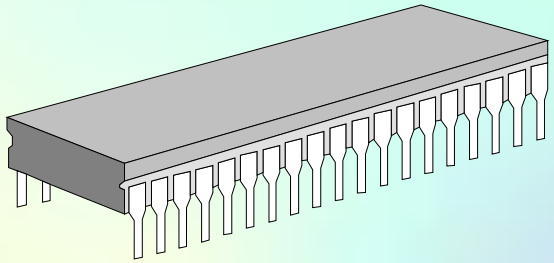
Dedutivos: são aqueles cuja conclusão é obtida como consequência das premissas;	Indutivos: são aqueles que apontam para uma verdade universal com base nos dados.

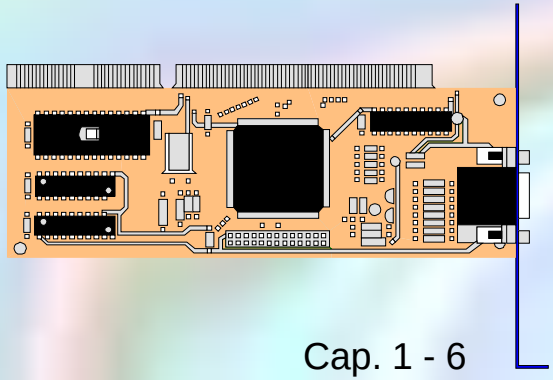
Raciocínio Lógico no Dia-a-Dia

- O homem utiliza-se do raciocínio lógico para a realização de suas atividades;
- Estabeleceu sequências adequadas para a realização de suas tarefas com sucesso:
  - Uma pessoa adulta, para tomar banho, primeiro tira a sua roupa para não molhá-la e também para estabelecer contato direto entre sua pele e a água.
  - Uma criança, desde pequenina, aprende que, para chupar uma bala, é preciso tirá-la da embalagem.

Lógica Aplicada à Informática

- Circuitos eletrônicos e portas lógicas;
- Softwares básicos e aplicativos;
- Algoritmos para a solução de problemas cada vez mais complexos;





Variáveis

- A lógica preocupa-se com a forma da construção do pensamento. Isso permite que se trabalhe com variáveis para que se possa aplicar o mesmo raciocínio a diferentes problemas. Por exemplo:

  Gerson é cientista.

  Todo cientista é estudioso.

  Logo, Gerson é estudioso.

  Substituindo as palavras ‘Gerson’ e ‘estudioso’ por A e B:

  A é cientista.

  Todo cientista é B.

  Logo, A é B.

Raciocínio Lógico

- O raciocínio lógico nos conduz a uma resposta que pode ser verdadeira ou falsa;
- Na construção de algoritmos para a solução de problemas computacionais, trabalha-se com esse tipo de raciocínio;
- As informações a serem analisadas são representadas por variáveis que posteriormente receberão valores. Essas variáveis representarão as premissas.

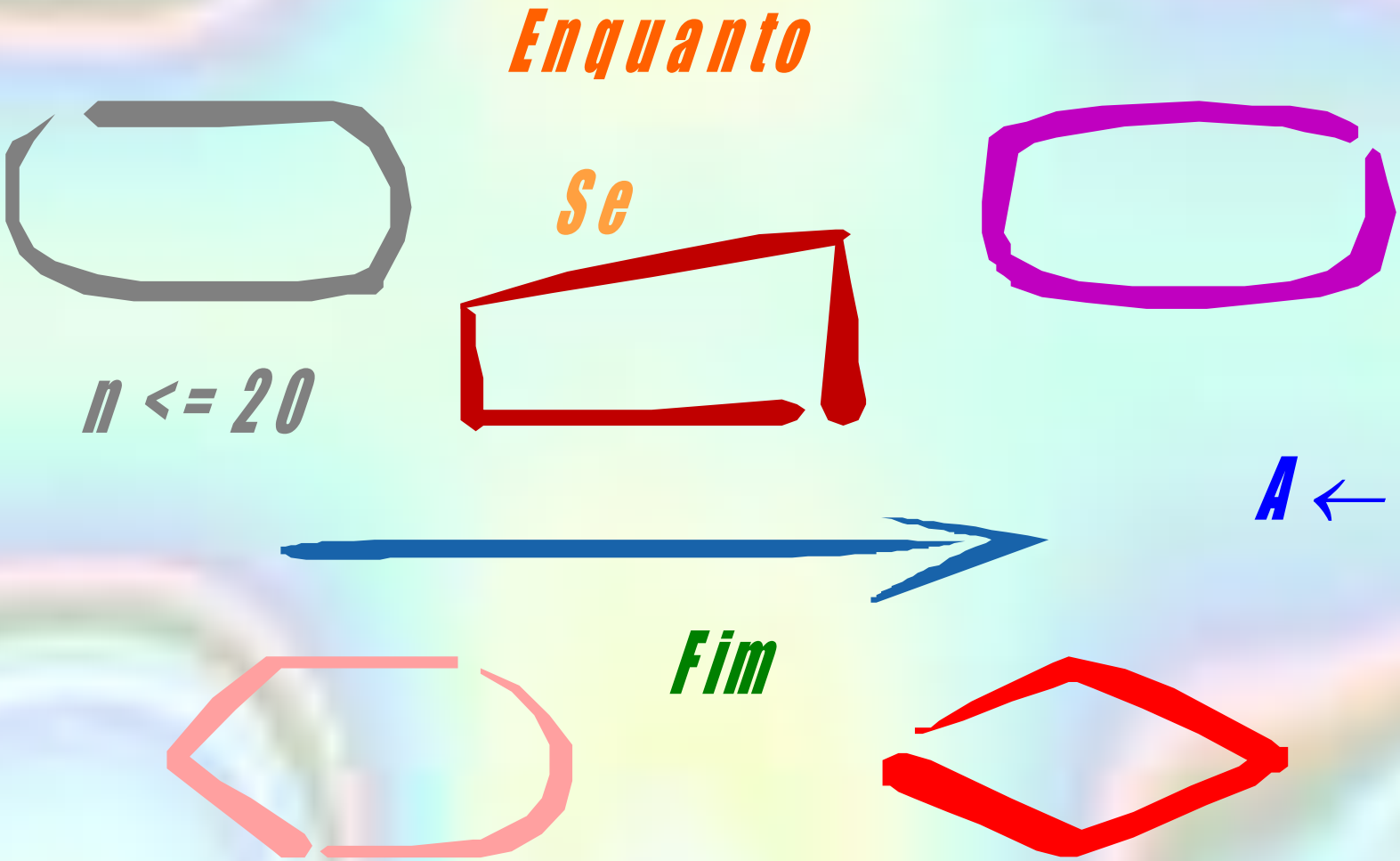
Raciocínio Lógico - Variáveis

Dados dois valores quaisquer, deseja-se saber qual é o maior:

- Os dois valores são representados pelas variáveis A e B;
- Analisa-se o problema e monta-se a seqüência para a verificação da questão: A é maior do que B?;
- Para que seja verificado o maior, deve-se fazer uma comparação, por exemplo: 7 é maior do que 19?
- Substituindo A por 7 e B por 19, obtém-se a resposta: Falso.

Portanto, as variáveis são utilizadas para a generalização de um problema.

Introdução a Algoritmos



Introdução a Algoritmos

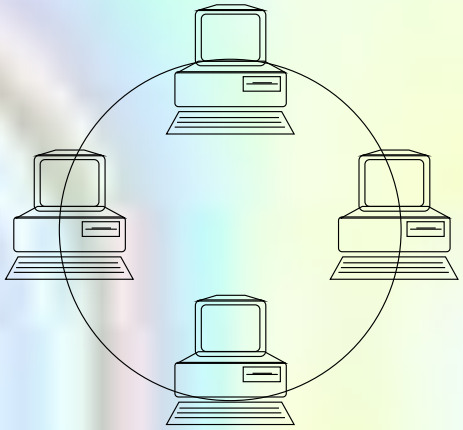
- **Objetivo:**
- Mostrar as aplicações dos algoritmos para a resolução de diferentes problemas;
- Especificar a importância de algoritmos para a resolução de problemas computacionais: abordar os conceitos de entrada, processamento e saída do ponto de vista computacional;
- Definir os tipos de algoritmos a serem utilizados (pseudocódigo e fluxograma).

Algoritmo

- Um algoritmo é uma sequência lógica de instruções que devem ser seguidas para a resolução de um problema ou para a execução de uma tarefa.
- Amplamente utilizados nas disciplinas ligadas à área de ciências exatas, tais como matemática, física, química e informática, entre outras.
- No dia-a-dia, as pessoas utilizam-se de algoritmos de maneira intuitiva:
  - A dona de casa utiliza-os para preparar um bolo;
  - Um motorista, para a troca de um pneu furado;
  - Um matemático, para resolver uma equação;
  - etc.

Algoritmos Aplicados à Computação

- Os algoritmos são amplamente utilizados na área da computação:
  - Elaboração de soluções voltadas à construção de interfaces: *softwares* e *hardware;*
  - Planejamento de redes;
  - Documentação de sistemas – descrevem as tarefas a serem realizadas pelos programas.



Tipos de Algoritmos

- Pseudocódigo: utiliza linguagem estruturada e se assemelha, na forma, a um programa escrito na linguagem de programação Pascal. Português estruturado;

- Fluxograma: utiliza-se de figuras geométricas para ilustrar os passos a serem seguidos na resolução dos problemas. Diagrama de Blocos. É bastante utilizado;

Pseudocódigo

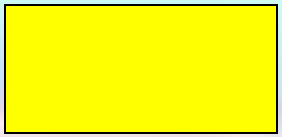
- É um tipo de algoritmo que utiliza uma linguagem flexível, intermediária entre a linguagem natural e a linguagem de programação;
- ‘Pseudocódigo’ significa ‘falso código’; o nome se deve à proximidade que existe entre um algoritmo escrito em pseudocódigo e a maneira pela qual um programa é escrito em uma linguagem de programação.

```
1  Algoritmo PrimeiroExemplo {
2      caracter nome, cargo;
3      inteiro idade, numPessoas, totalPessoas;
4      real salario;
5      numPessoas = 1;
6      totalPessoas = 0;
7      enquanto (numPessoas <= 50)
8      {
9          escreva ("Digite nome, idade, cargo e salario: ");
10         leia (nome, idade, cargo, salario);
11         se ((idade <= 30) e (salario >= 3000))
12         {
13             totalPessoas = totalPessoas + 1;
14         }
15         numPessoas = numPessoas + 1;
16     }
17     escreva("Total de pessoas que atendem à condição:", totalPessoas)
18 }
```

Fluxograma – Simbologia

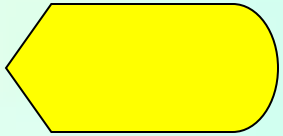
- Cada instrução ou ação a ser executada deve ser representada por meio de um símbolo gráfico.

Terminal: representa o início e o final do fluxograma.

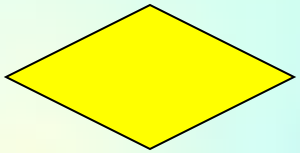


Processamento: representa a execução de operações ou ações.

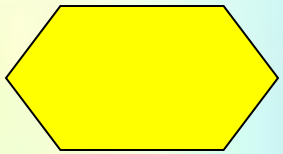
Teclado: representa a entrada de dados para as variáveis por meio do teclado.



Vídeo: representa a saída de informações por meio do monitor de vídeo.



Decisão: representa uma ação lógica que resultará na escolha de uma das seqüências de instruções.



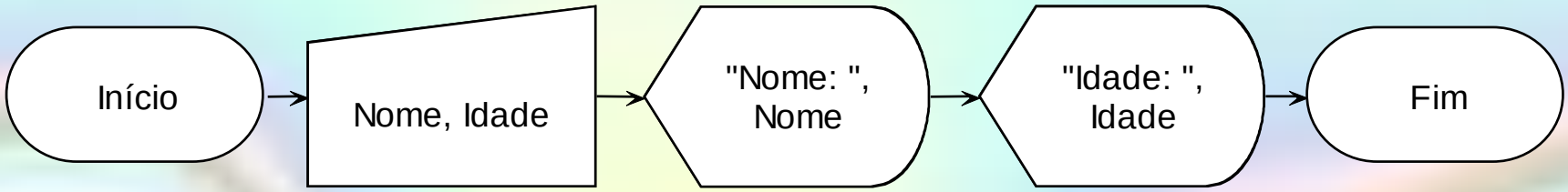
Preparação: representa uma ação de preparação para o processamento.

Fluxograma – Simbologia

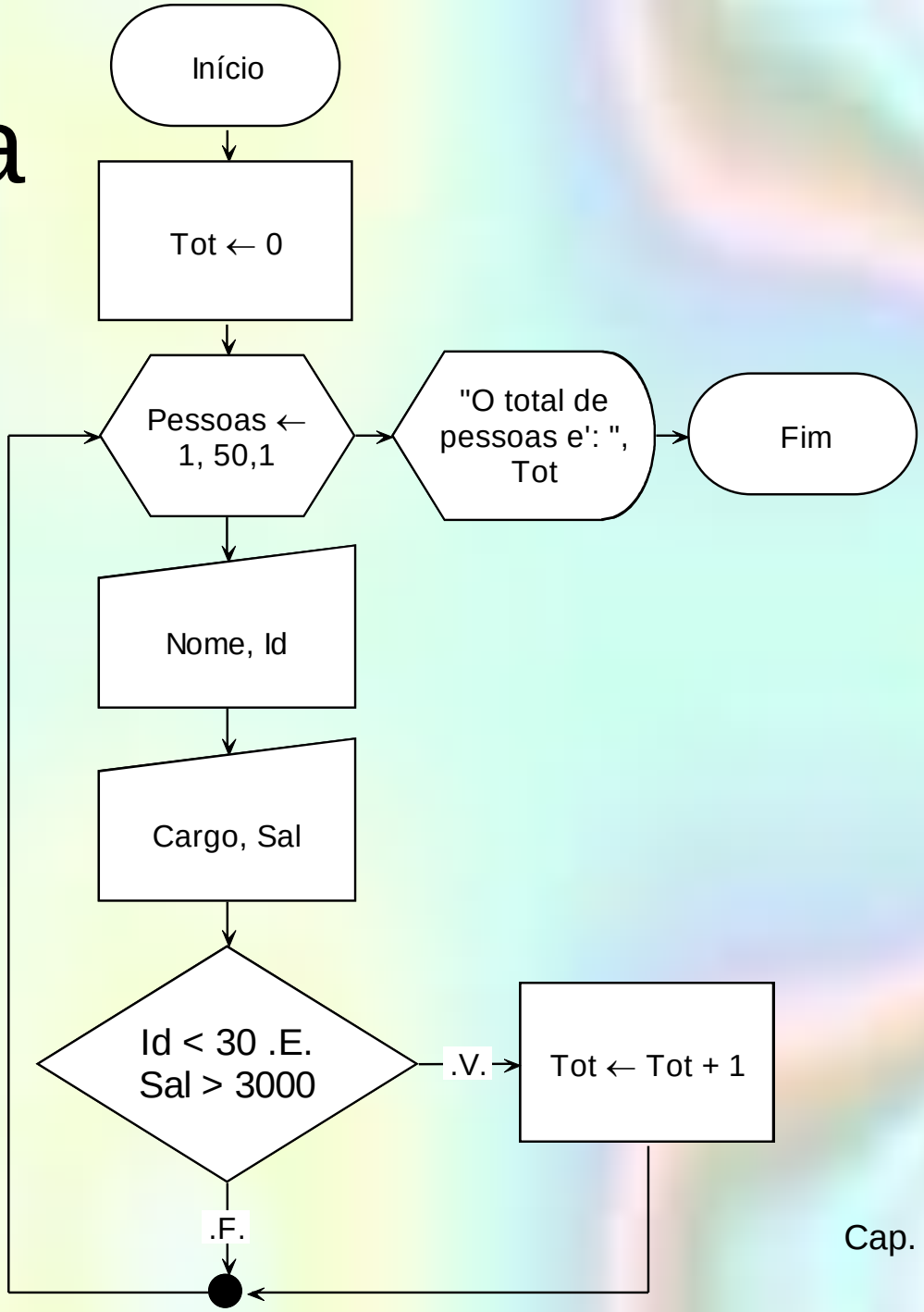
Conector: utilizado para interligar partes do fluxograma ou para desviar o fluxo corrente para um determinado trecho do fluxograma.

Conector de Páginas: utilizado para interligar partes do fluxograma em páginas distintas.

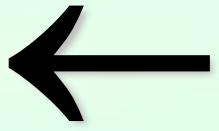
Seta de orientação do fluxo.

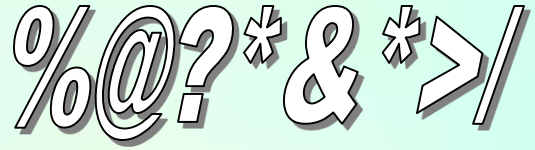


Fluxograma Exemplo



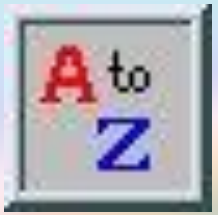
Conceitos Básicos sobre Algoritmos





AF10

01011100



Conceitos Básicos sobre Algoritmos

Objetivo:

- Mostrar os tipos de dados básicos e seus desdobramentos na linguagem Java;
- Definir o conceito, a aplicação e a identificação de variáveis e constantes;
- Demonstrar o uso dos operadores de atribuição, aritméticos, relacionais e lógicos;
- Exemplificar a construção de expressões de atribuição, aritméticas e lógicas.
- Mostrar a ordem de precedência matemática utilizada na resolução de problemas.

Dados e Variáveis

- Os dados são, na verdade, os valores que serão utilizados na resolução de um problema;
  - Esses valores podem ser fornecidos pelo usuário do programa;
  - Podem ser originados a partir de processamentos (cálculos);
  - Ou a partir de arquivos, bancos de dados ou outros programas.
- Os dados são armazenados temporariamente em variáveis para que sejam processados de acordo com as especificações do algoritmo.

Dados e Variáveis

- Os tipos de dados são definidos a partir dos tipos primitivos — características dos computadores;
- Como os dados manipulados são armazenados na memória, esses tipos seguem as características de formato e o espaço disponível nessa memória;
- São organizados em bits e bytes e suas combinações:
  - Exemplo para representar um número inteiro:

    usar dois bytes ou 16 bits;

    resultando em 2^{16} combinações = 65536 possibilidades;

    Lembrando que os números poderiam assumir valores negativos e positivos nessa faixa, teríamos representações que iriam de

    –32768 a 32767.

Tipos de Dados

- Definir o tipo de dado de uma variável é uma questão de grande importância para garantir a resolução do problema;
- É necessário que se tenha conhecimento prévio do tipo de informação (dado) que será utilizado para resolver o problema proposto;
- Nos algoritmos, utilizamos os tipos de dados primitivos (caracter, inteiro, real e lógico);
- Nos programas, para esses tipos de dados, existem desdobramentos adequados a cada linguagem de implementação.

Primitivos		Específicos para linguagem de Programação Java	
Tipos de dados	**Definição**	**Tipos de dados**	**Capacidade de armazenamento na memória do computador, de acordo com a linguagem Java**
Literal – também conhecido como texto ou caractere	Poderá receber letras, números e símbolos. Obs.: Os números armazenados em uma variável cujo tipo de dado é literal não poderão ser utilizados para cálculos.	char	16 bits – armazena Unicodes **Nota:** Também é possível armazenar dados do tipo literal na Classe String.
Inteiro	Poderá receber números inteiros positivos ou negativos	byte	8 bits de (-128) até (127)
		short	16 bits de (-32.768) até 32.767
		int	32 bits de (-2.147.483.648) até (2.147.483.647)
		long	64 bits de (-9.223.372.036.854.775.808) até (9.223.372.036.854.775.807)
Real – também conhecido como ponto flutuante	Poderá receber números reais, isto é, com casas decimais, positivos ou negativos	float	32 bits de (–3,4E-38) até (-3,4E+38)
		double	64 bits de (-1,7E-308) até (+1,7E+308)
Lógico – também conhecido como booleano	Poderá receber verdadeiro (1) ou falso (0)	boolean	8 bits – em Java pode-se armazenar true ou false

Tipos Construídos

- Nos algoritmos, assim como nas linguagens de programação, existe a possibilidade de criar outros tipos de dados, chamados tipos construídos.

```
Algoritmo Exemplo_Registro
    Tipo
      Reg_paciente = registro
                  Nome: literal
                  Idade: inteiro
                  Peso: real
      fim_registro
    Var
      Paciente: Reg_paciente
...
```

Identificação das Variáveis

- Nos algoritmos, toda variável deve ser identificada, recebendo um nome ou identificador;
- O nome de uma variável deve ser único e deve estar de acordo com algumas regras:
  - não utilizar espaços entre as letras. Por exemplo, em nome do cliente, o correto é nome_do_cliente ou nomecliente;
  - não iniciar o nome da variável com números. Por exemplo, em 2valor, o correto é valor2;
  - não utilizar palavras reservadas. Exemplo: se - palavra que representa uma condição ou teste lógico; var - palavra que representa a área de declaração de variáveis, entre outras;
  - não utilizar caracteres especiais, como acentos, símbolos (? / : @ # etc), ç, entre outros;
  - ser sucinto e utilizar nomes coerentes.

Identificadores de Variáveis - Java

- Java e os nomes para as variáveis — “case-sensitive”. Nomes com letras maiúsculas são diferenciados de nomes com letras minúsculas. Exemplo: *NomeCliente* é diferente de *nomecliente,* que é igualmente diferente de *nomeCliente*.
  - nomes devem começar com uma letra, um caractere ‘sublinhado’ ou ‘*underline*’ (_) ou o símbolo cifrão ($). Os caracteres subsequentes podem também incluir números;
  - não utilizar caracteres especiais, como acentos, símbolos (? / : @ # etc), ç, entre outros, exceto os acima citados;
  - as letras podem ser maiúsculas ou minúsculas;
  - não podem ser utilizadas palavras reservadas, como: final, float, for, int, etc.

Constantes

- São valores que não sofrem alterações ao longo do desenvolvimento do algoritmo ou da execução do programa;

  pi = 3.1415;

  perímetro = 2 * pi * raio;

  no exemplo, o valor 3.1415 é atribuído à constante *pi* e permanecerá fixo até o final da execução.

- Em Java, uma constante é uma variável declarada com o modificador *final.* Por exemplo:

  final float pi = 3.1415 f;

Operadores

- Os operadores são utilizados para representar expressões de cálculos, comparação, condição e atribuição;
- Operadores de atribuição: utilizados para expressar o armazenamento de um valor em uma variável;
  - nome ← “Fulano de tal” nome = “ Fulano de tal “
- Operadores aritméticos: utilizados para a realização dos diversos cálculos matemáticos;
  - resultado ← a + 5 resultado = a + 5

Operadores

- Operadores relacionais: utilizados para estabelecer uma relação de comparação entre valores ou expressões. O resultado dessa comparação é sempre um valor lógico (booleano) verdadeiro ou falso;
  - c ← a > b c = a > b
- Operadores lógicos: utilizados para concatenar ou associar expressões que estabelecem uma relação de comparação entre valores. O resultado destas expressões é sempre um valor lógico (booleano) verdadeiro ou falso.
  - a = 5 .e. b <> 9 a == 5 && b != 9

Precedência dos Operadores

- As linguagens de programação e a precedência dos operadores, quando mais de um operador é usado em uma expressão;
- No caso de não haver precedência, a expressão é avaliada da esquerda para a direita;
  - A = B < 8 .e. C = 3
  - Combinação de duas expressões de comparação: B < 8 e C = 3.
  - Se ambas resultarem em verdadeiro, o valor verdadeiro será atribuído a variável A. Em qualquer outra circunstância, o valor falso será atribuído a A.
- Precedência entre os operadores relacionais e os lógicos;
- Precedência entre operadores de tipos diferentes e operadores do mesmo tipo.

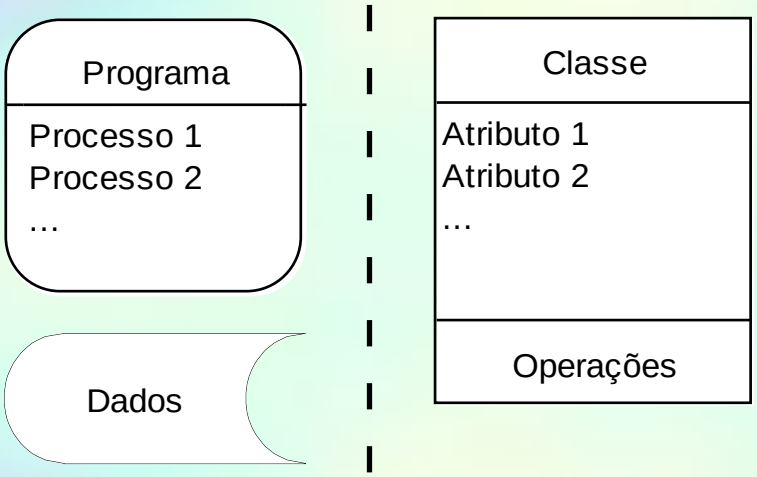
Tabela Verdade

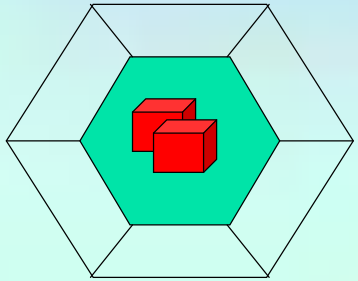
- Expressa o conjunto de possibilidades existentes para a combinação de variáveis ou expressões e operadores lógicos. Exemplo de combinação entre variáveis: A = 5 .e. B != 9

			Operador		
			&&	||	!
Expressão	A = 5	B != 9	A = 5 && B != 9	A = 5 || B != 9	!A = 5
Resultados possíveis	.v.	.v.	.v.	.v.	.f.
	.v.	.f.	.f.	.v.	.f.
	.f.	.v.	.f.	.v.	.v.
	.f.	.f.	.f.	.f.	.v.

Conceitos de Programação

Enfoque Tradicional X Orientado a Objeto





```
/* funcao.c */
main()
{
        int x, y, r;
        printf("Digite dois numeros: ");
        scanf("%d %d",&x, &y);
```

```
class Robo
{
        private String nome;
        private int serie;
        private String data;
        public int bat;
```

Conceitos de Programação

- **Objetivo**:
- Abordar os principais paradigmas em programação;
- Os diferentes tipos de linguagens de programação e suas aplicações;
- O pseudocódigo como uma maneira estruturada de representação das soluções de problemas e a linguagem de programação Java;
- Os principais conceitos de programação orientada a objetos.

Introdução à Programação

- Um programa é um conjunto de instruções que dizem ao computador o que deve ser feito;
- Os diferentes tipos de linguagens de programação e suas aplicações;
- Os níveis das linguagens;
  - Linguagens de programação que atuam diretamente no hardware da máquina, movimentando dados e acionando dispositivos ligados ao computador (baixo nível);
  - Linguagens de alto nível, como Pascal, C, C++ e Java, utilizadas pelos programadores (código-fonte);
  - Tradução para outros programas de baixo nível — processo chamado de Compilação ou Interpretação.
- As linguagens de programação, ditas de alto nível, e o objetivo de aproximar a programação da linguagem humana.

Introdução à Programação

- A linguagem de programação, como qualquer linguagem, é formada por palavras que são agrupadas em frases para produzir um determinado significado;
- Palavras-chave: as palavras de uma linguagem de programação;
- Estruturas de programação: as frases criadas com essas palavras-chave;
- Um programa é constituído de *palavras-chave* e *estruturas de programação* definidas segundo as regras dessa linguagem;

Introdução à Programação

- A exemplo da linguagem usada em nossa comunicação no dia-a-dia, a linguagem de programação possui uma sintaxe, definida por essas regras;
- Por que existem tantos tipos de linguagem?
- Uma linguagem é melhor do que outra?
- A resposta para essas perguntas está no objetivo para o qual elas foram criadas:
  - Uma linguagem pode ser melhor para a execução de cálculos matemáticos complexos, com aplicações na área científica;
  - Outra pode ser melhor para processar uma grande quantidade de dados submetidos a operações simples, com aplicações na área financeira;
  - Outras exigem uma interface elaborada e fácil interação com o usuário.

Algumas Linguagens

- **Pascal**
  - É uma linguagem de alto nível poderosa e eficientemente estruturada. Criada para ser uma ferramenta educacional pela simplicidade de sua sintaxe.

WRITE (“Algoritmos e Estruturas de Dados”);

- **C**
  - Linguagem estruturada utilizada até pouco tempo para o desenvolvimento de aplicações comerciais. Ultimamente, tem grande aplicação no desenvolvimento de software básico e aplicações com forte interação com o hardware.

Printf (“Algoritmos e Estruturas de Dados”);

Algumas Linguagens

- **C++**
  - Linguagem de alto nível orientada a objetos; uma evolução do C que preserva seus princípios de eficiência e facilidade de programação.

```
cout << “Algoritmos e Estruturas de Dados”;
```

- **Java**
  - Linguagem orientada a objetos de fácil programação e larga utilização no mercado. Amplamente utilizada em aplicações de processamento distribuído e para a Internet.

```
System.out.println("Algoritmos e Estruturas de Dados").
```

Programação Linear

- Programação Linear: programas que, na sua execução, obedecem a uma sequência de passos executados consecutivamente, com início e fim específicos;
- Linguagens mais antigas utilizavam-se desse princípio, como o Basic, que numerava as linhas de código uma a uma. Desvios eram executados apontando-se para a linha desejada;
- Programas lineares podem ser gerados utilizando-se linguagens estruturadas ou orientadas a objetos;
- A desvantagem da programação linear é a complexidade. Programas lineares extensos são difíceis de serem desenvolvidos e até compreendidos.

Programação Estruturada

- Baseia-se no princípio de dividir as tarefas a serem realizadas em etapas, executando-as uma por vez, até que todo o trabalho tenha sido realizado;
- Programa monolítico X Programa estruturado;
- Programação Estruturada e Modularização;
- Procedimentos ou Funções: blocos de programa que executam determinada tarefa:
  - Procedimentos: podem receber valores, mas não retornam outros valores como resultado;
  - Função: retorna os valores resultantes das operações que realizou.

AtualizarDados()

soma(x, y)

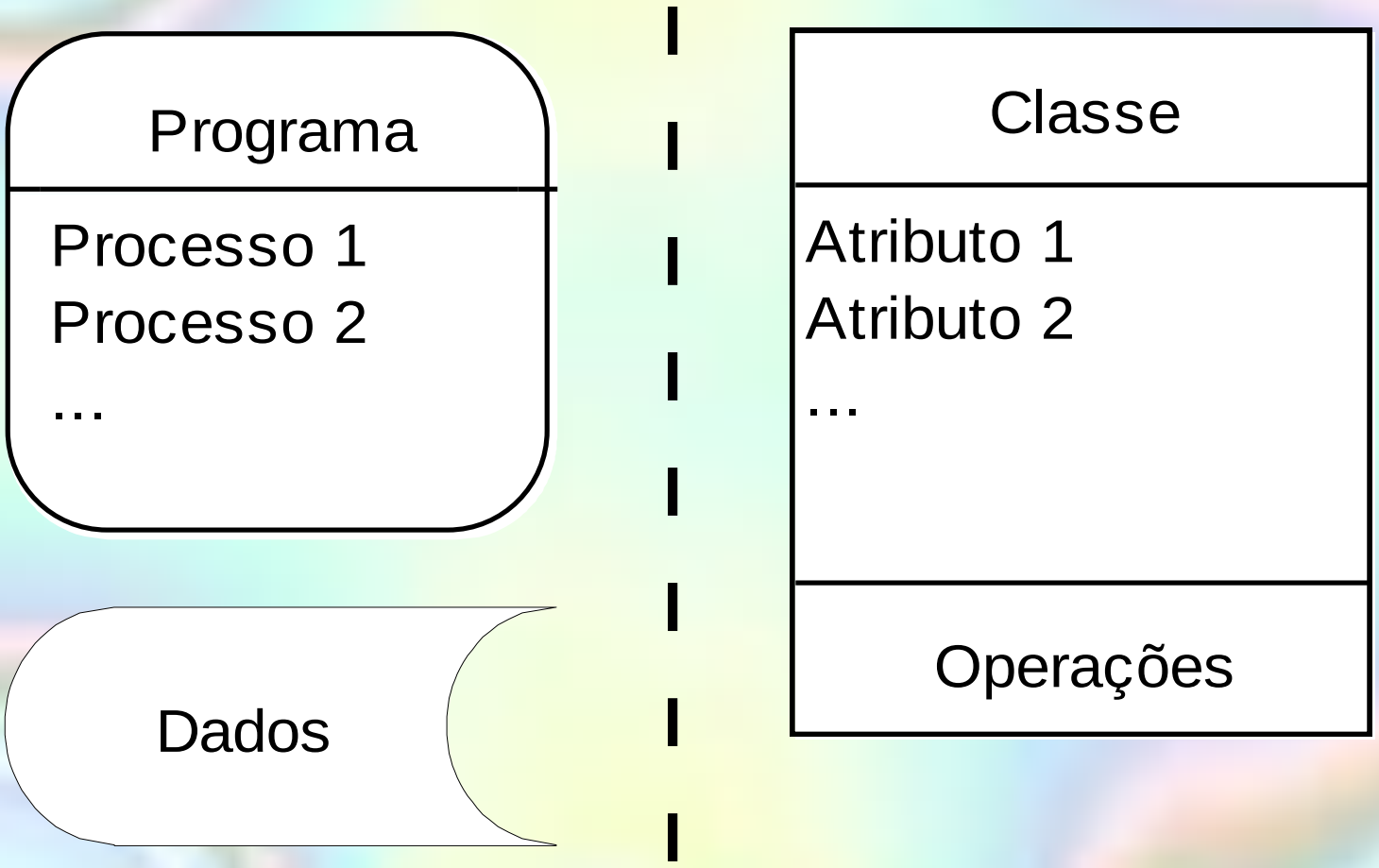
Programação Estruturada - Exemplo

```
/* funcao.c */
main()
{
        int x, y, r;
        printf("Digite dois numeros: ");
        scanf("%d %d",&x, &y);
        r = soma(x, y);
        printf("A soma dos numeros e”: %d", r);
}
/* soma() */
/* retorna a soma de dois numeros */
soma(j, k)
int j, k;
{
        return(j+k);
}
```

Programação Orientada a Objetos

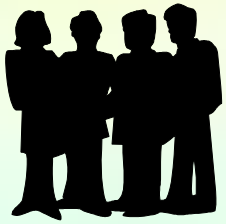
- O enfoque tradicional: um sistema é um conjunto de programas inter-relacionados que atuam sobre um determinado conjunto de dados que se deseja manipular de alguma forma para obter os resultados desejados;
- O enfoque da modelagem de sistemas por objetos: procura enxergar o mundo como um conjunto de objetos que interagem entre si, apresentam características e comportamento próprios, representados pelos seus atributos e suas operações. Os atributos estão relacionados aos dados, e as operações, aos processos que o objeto executa.

Enfoque Tradicional X Orientado a Objeto

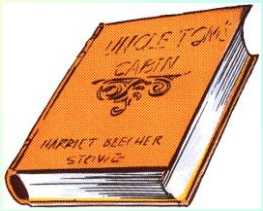


O que é um Objeto?

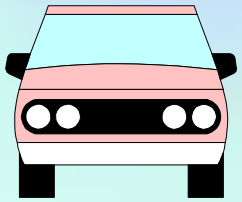
Tangíveis



Pessoas

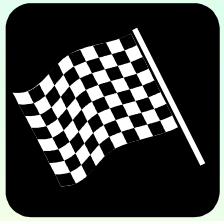


Livro

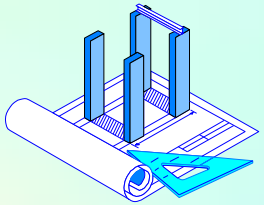


Automóvel

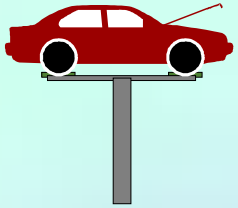
Incidente



Competição

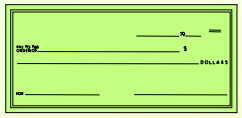


Projeto

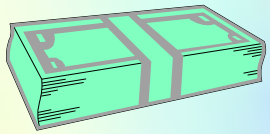


Conserto

Interação



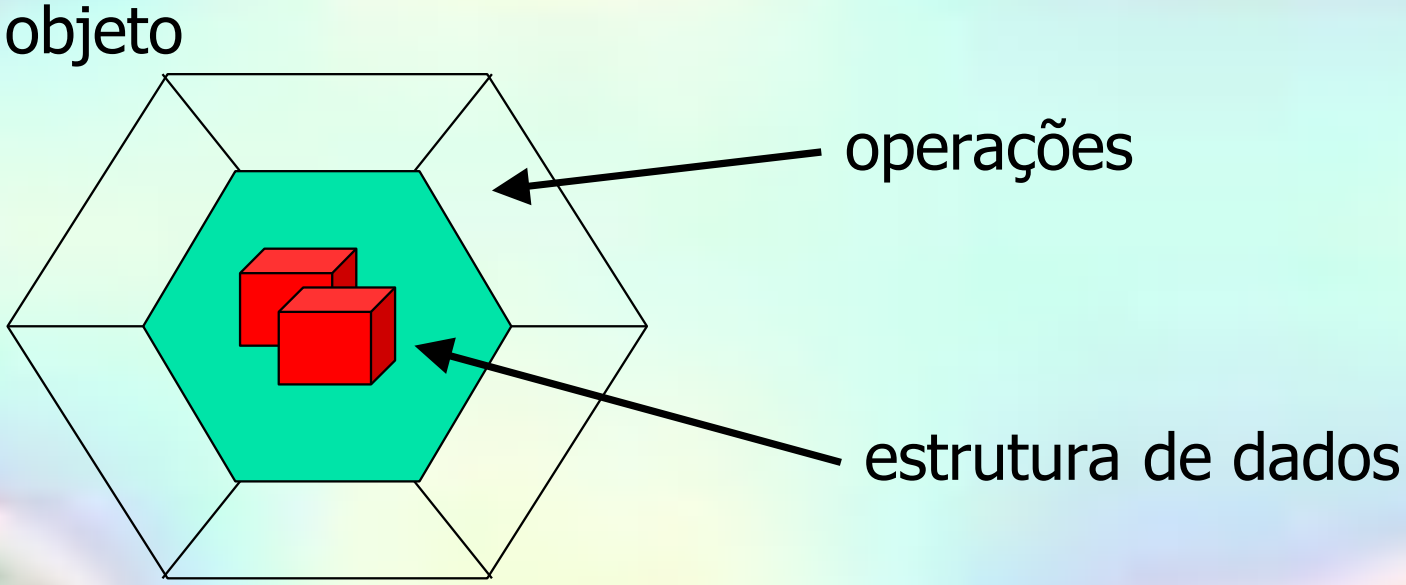
Transação



Saque

Venda

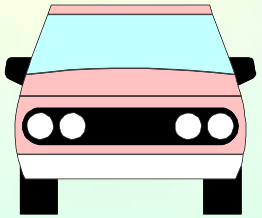
Como Visualizar um Objeto?



Classes

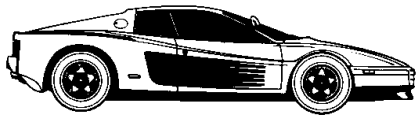
Generalização

Especialização



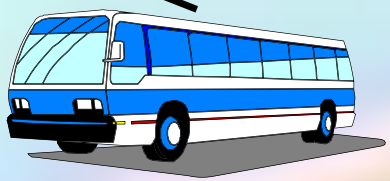
Veículos

Utilitário



Esporte

Passeio



Passageiros

Instâncias de Objetos

Considere um sistema de uma revendedora de veículos. Cada novo veículo adquirido pela revendedora seria cadastrado no sistema, criando um novo objeto dessa classe, que será chamada de **instância de objeto**, conforme o esquema abaixo:

Classe	Sub classe	Sub classe	Instância	Instância
Veículos	Passeio	Sedã	marca: Opel modelo: Fire ano: 2002 potência: 195cv. eixos: 2 carga: 1500Kg.	marca: Thunderbird modelo: Hatch ano: 2000 potência: 250cv. eixos: 2 carga: 1800Kg.

Herança

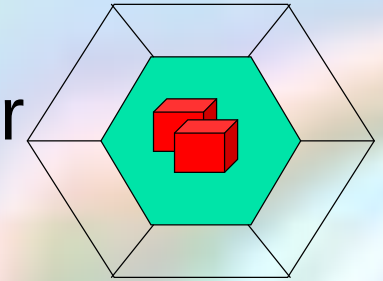
Uma classe é constituída de objetos com atributos e operações semelhantes;

A herança nada mais é do que a implementação da generalização; é o compartilhamento de atributos e operações entre classes com base em um relacionamento hierárquico;

Quando se cria uma nova instância de um objeto, dizemos, em orientação a objeto, que esse novo objeto herda os atributos e as operações de sua classe.

Encapsulamento

- O encapsulamento, também chamado de ocultamento de informações, consiste na separação entre os aspectos externos de um objeto, acessíveis por outros objetos, e os detalhes internos da implementação daquele objeto, que ficam ocultos dos demais objetos;
- O encapsulamento impede que um programa se torne tão interdependente que uma pequena modificação possa provocar grandes efeitos que se propaguem por todo o sistema.



Bibliografia