**Anotações e Estudo T.I**

O que é e como funciona:

* **ServiceNow:** Companhia norte-americana de computação em nuvem que fornece softwares de gerenciamento de serviços como um serviço por meio de uma plataforma de nuvem corporativa de serviços de dados único. Especializada em gerenciamento de serviços de TI (ITSM), gerenciamento de operações de TI (ITOM) e gerenciamento de negócios de TI (ITBM).
* **SalesForce:** É uma empresa de softwares que foca na solução de gerenciamento de relacionamento para aproximar empresas e pessoas. É uma plataforma de CRM integrada que oferece a todos os departamentos uma visão única e compartilhada de cada cliente.
* **Cloud:** Computação em nuvem é o conceito que se aplica ao acesso de um conteúdo que está num ambiente externo, que não nos pertence, por meio da internet. Entrega vários tipos de serviços, não somente em T.I.
* **Java:** Linguagem de programação orientada a objetos super versátil.

**Introdução a Lógica de Programação**

Muito relacionado a computação e á sua história. Sempre houve lógica, ela sempre esteve presente, mas apenas conceituada e formalizada por Aristóteles na Grécia antiga. Se baseia nos princípios racionais e nas argumentações válidas. Existem vários tipos de lógica: Matemática aristotélica, simbólica, etc. Apenas nos concentraremos na algébrica e formal.

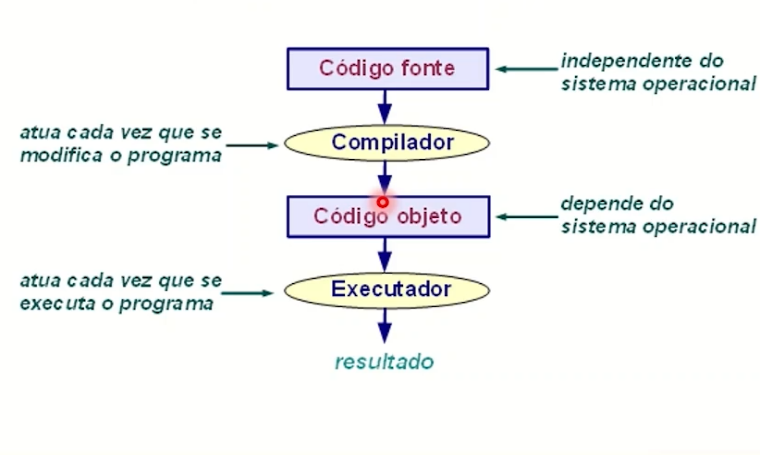
Para existir lógica não precisa haver fatos verdadeiros, apenas um ponto de inicio e um ponto final que faça sentido, ou seja, a lógica consiste na construção racional e coerente do raciocínio.

**Lógica Algébrica**

Veio á tona a partir dos trabalhos de George Boole e Augustus de Morgan e possibilitou o aprofundamento na programação. Sendo assim a Álgebra Booleana está relacionada tanto na construção de softwares quanto de Hardwares

**Linguagens de Programação Compiladas e Interpretadas**

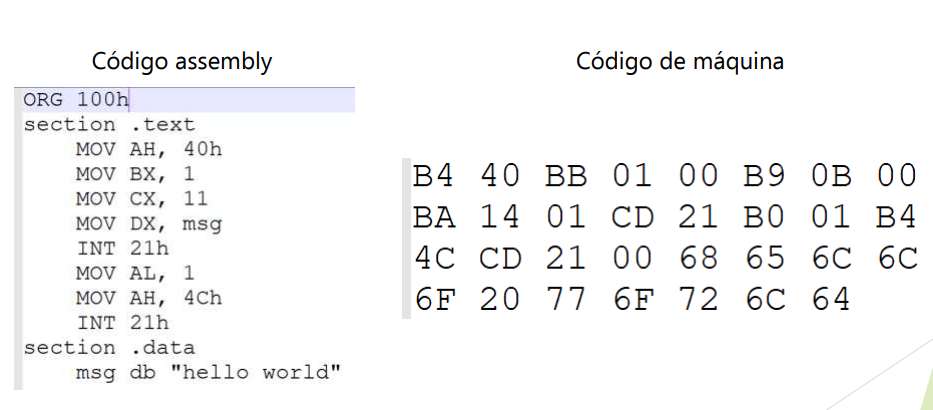
Nas compiladas utilizamos um programa de compilação para traduzir um código fonte de uma linguagem para outra para que o sistema operacional a interprete.



**Assembly:** É o nome dado ao código compilado e simples, se torna outra linguagem. Nela sabemos exatamente o que o processador está fazendo. Não é muito fácil de entender, mas é exatamente o que roda no cérebro do pc.

**Assembler:** É o nosso executador, que lê o arquivo contendo os comandos simples do assembly e os transforma em linguagem de máquina.

**Linguagem de máquina:** É a única coisa que o processador consegue interpretar, são apenas códigos, não dá para ser lido por humanos (Normais).



Nas interpretadas apenas utilizamos um interpretador como um navegador para rodar o conteúdo por exemplo.

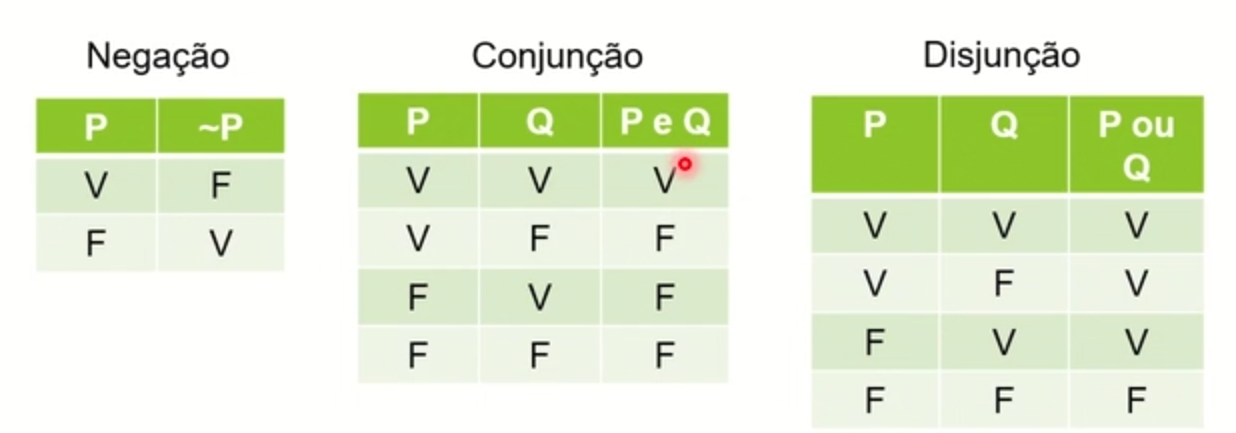
Resumo: Nas compiladas a tradução é feita apenas uma vez. Já nas interpretadas a tradução é feita sempre quando se executa o software com o conteúdo.

**Lógica na Solução de Problemas**

**Algoritmo:** Sequência de ações que obedecem a uma ordem que possuem de um estado inicial durando determinado tempo resultando em estados esperados e bem definidos. Fazemos isso até inconscientemente no dia a dia, como comer, respirar ou tomar banho.

**Lógica Formal:** Ferramenta que nos ajuda a desenvolver uma cadeia de pensamentos reconhecendo possíveis falhas, identifica raciocínios válidos e não válidos.

**Álgebra Booleana:** Entrando na programação a importância da álgebra booleana está na sua utilização de símbolos e operações algébricas para representar proposições e suas inter-relações.



Algoritmos

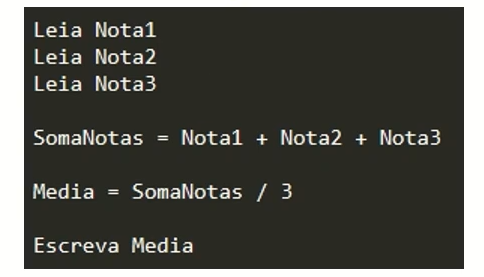
Conjunto de ações perfeitamente definidos com números finitos de etapas antecipando falhas para concluir seu objetivo. DEVE SER CLARO E OBJETIVO.

Estes são os passos que devemos seguir para aplica-los na programação:

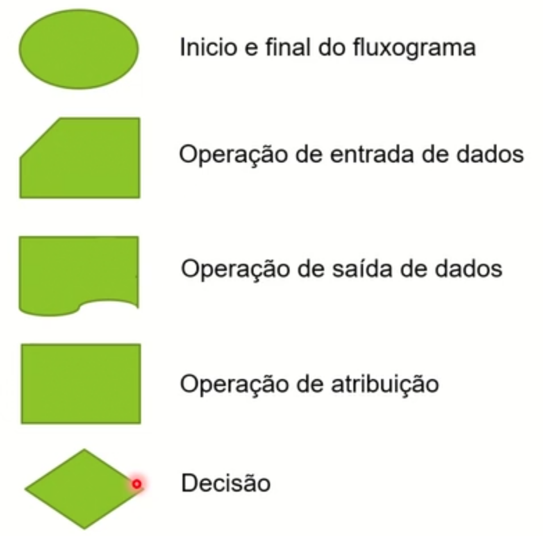


pseudocódigo: Descrição como algoritmo.

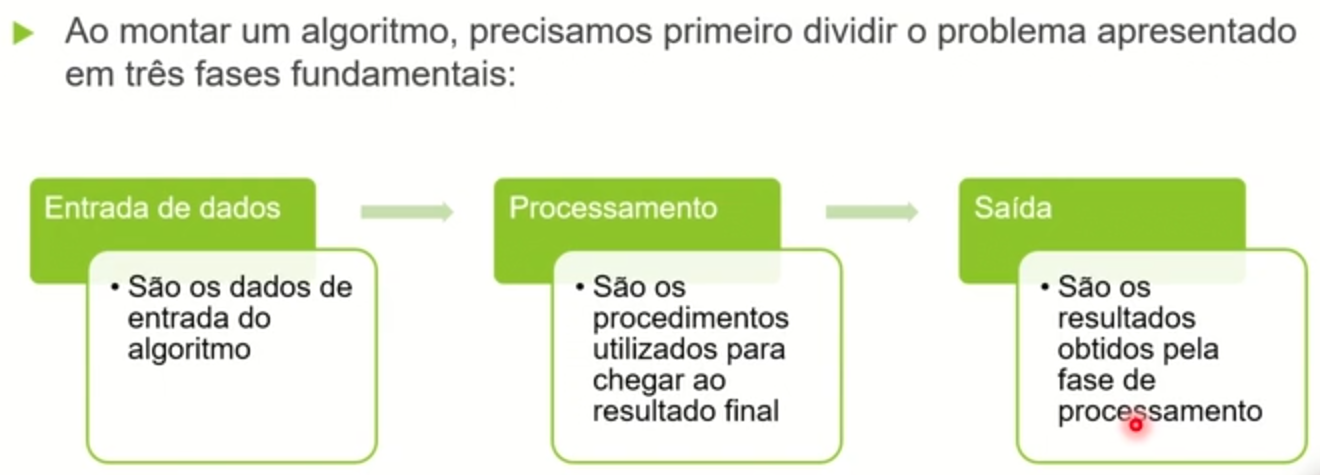
Ex - Somar as 3 notas de 3 provas diferentes de um aluno a fim de conseguir uma média.



Fluxograma: Representação gráfica como algoritmo.



Construção do Algoritmo:



**Módulo 1**

**Operadores**

São os símbolos ou palavras que utilizamos na lógica de programação.

Sendo eles:

* **Literal ou String:** Só pode ser somado, aceita vários caracteres;
* **Carácter ou Char:** Só pode ser somado, aceita apenas um caracter;
* **Inteiro:** Pode ser somado, subtraído e multiplicado;
* **Real:** Pode ser divido;
* **Lógico:** Diz ser verdadeiro ou falso.
* **^:** Potência.
* **Escolha:** Para determinar alternativas simples.
* **Fimescolha:** Finaliza a escolha.
* **Caso:** Para atribuir uma situação na escolha.
* **Casonao:** Para atribuir nova situação caso a primeira não funcione (Bem parecido com se e senao).
* **Div:** Apresenta somente o número inteiro da divisão.
* **Mod ou %:** Apresenta o restante da divisão.
* **Enquanto ou WHILE:** Calcula ou executa determinada situação automaticamente em loop, se não houver uma saída o while continuará infinitamente.
* **Fimenquanto:** Finaliza o enquanto.
* **Repita:** Só funciona quando se pode enxergar um fim e repetirá a sentença até ter a resposta desejada como saída.
* **Ate:** Finaliza o repita (Não conta como flag de saída!).
* **Para:** Este operador pode substituir o enquanto e o repita em algumas situações, deixando o código mais limpo visualmente, muito usado para vetores e banco de dados.
* **Passo:** Ao atribuir a tag "passo" no operador “para”, você determina uma regra de multiplicação, de em quantos números serão exibidos.
* **Aleatório:** Sorteia um valor aleatório na execução
* **Comparadores de operação:**
* **Se:** Compara;
* **Então:** Atribui;
* **Senao:** Substitui o se caso não funcione;
* **Fimse:** Finaliza a comparação.
* **Tabela Verdade:** Podem diminuir as comparações!
* **E:** Apenas válido quando toda a afirmação é verdadeira.
* **Ou:** Válido mesmo quando uma parte da afirmação é falsa, mas a outra verdadeira.

**Módulo 2**

**Procedimentos, Banco de Dados e Orientação de Objetos**

**Debug:** É analisar um código linha por linha, facilita para encontrar um erro. Cada linguagem de programação tem uma forma de fazer.

**Vetores (Array):** É como um baú capaz de armazenar e executar em apenas uma variável várias outras, quantas quiser ou precisar.

**Matrizes: Tipos de vetores de alta dimensão.**

**Procedimentos:** Método para organizar variáveis. Processa de acordo com dados de entrada.

**Função:** Processa com resposta em retorno.

**Banco de dados**

SQL: Stuctured Query Language, ou Linguagem de Consulta Estruturada, é a linguagem de pesquisa declarativo padrão para banco de dados relacional. Muitas das características originais do SQL foram inspiradas na álgebra relacional.

XAMPP: É um pacote com os principais servidores de código aberto do mercado, incluindo FTP, banco de dados MySQL e Apache com suporte as linguagens PHP e Perl.

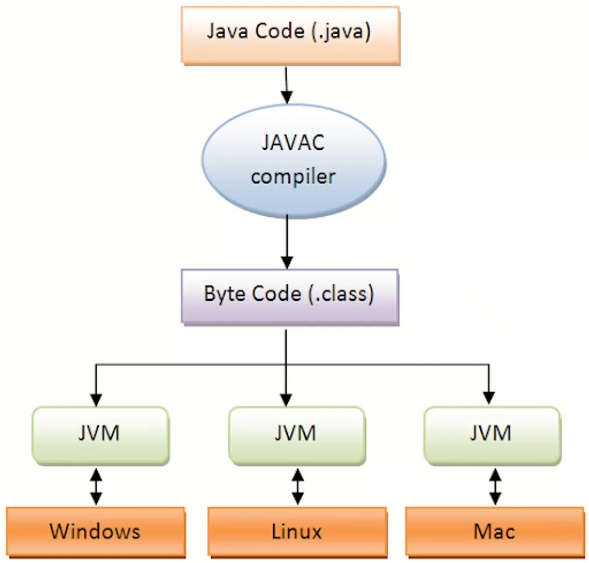
Primary Key: Transforma um item em algo único no sistema e fácil de achar pelo seu ID.

Inner Join: Permite a junção entre duas ou mais tabelas, desde que tenham a ver uma com a outra

**Java**

Criado em 1995 por uma equipe chefiada por James Gosling na empresa SunMicrosystem

de fácil compilação e executável pela maioria dos sistemas operacionais.



Operadores no JAVA:

* Primitivas: São simples e se limitam a valores em bits.
* Int: Pode ser somado subtraído e multiplicado;
* Float: Armazena valores numéricos em ponto flutuante (que representa números com casas decimais, tanto negativos quanto positivos) de precisão simples;
* Double: Armazena números com casas decimais;
* Char: Armazena apenas um caractere, nele utiliza-se aspas simples (‘’);
* Byte: Parecido com o int;
* Boolean: Armazena valores lógicos (True ou False).
* Não Primitivas: São aquelas que não armazenam valores em bits e são mais complexas e abstratas.
* String: Só pode ser somado, aceita vários caracteres até quando não são números;

Operadores Lógicos:

* &&: Apenas válido quando toda a afirmação é verdadeira;
* ||: Válido mesmo quando uma parte da afirmação é falsa, mas a outra verdadeira.
* !: Negação

Operadores Relacionais:

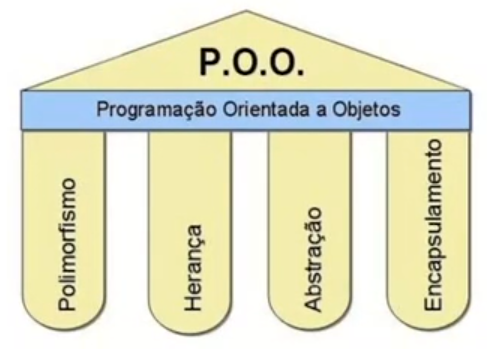
* >: Maior;
* >=: Maior ou igual;
* <: Menor;
* <=: Menor ou igual;
* ==: Compara igualdade;
* !=: Diferente.
* If/else: Avalia expressões como verdadeira ou falsa de acordo com uma situação predeterminada pelo usuário, executando uma ou outra rotina;
* Switch: Para definir escolhas;
* Case: Para definir um resultado para determinada escolha;
* Break: Para finalizar o Case.
* Default: Será executado em último caso quando nenhum dos Case for.

Outros Operadores:

* For: Utiliza-se quando se sabe até onde quer chegar.
* While: Repete uma instrução até que uma expressão booleana seja verdadeira.
* True: O while ficará em loop infinito.

**Orientação a Objeto**

Veio do SmallTalk em 1960. É uma forma de resolver problemas de uma maneira não mais estruturada pensando no problema e seus agentes como real, trazendo isso para o algoritmo. Esses são os seus 4 princípios:

****

Polimorfismo: Princípio no qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que tem a mesma identificação (assinatura) mas comportamentos distintos, especializados par cada classe derivada, usando para tanto uma referência a um objeto do tipo de superclasse. Resumidamente é como uma palavra sinônima de outra, que pode se aplicar a mesma situação das palavras sinônimas e a outras aplicações distintas.

Herança: Permite criar uma nova classe a partir de uma já existente. Também chamadas de subclasses.

Abstração: Reconhecer uma entidade real e passa-la para um algoritmo, extraindo suas informações necessárias para resolver um problema.

Encapsulamento: Vinculado ao conceito de segurança, o encapsulamento tem a ver com proteger uma informação ou esconde-la.

Conceitos básicos de P.O.O:

Encapsulamento: Garante a segurança das informações do objeto

Classe: Agrupamento de objetos. Ex: carro ou pessoa;

Objeto: Entidade abstrata ou concreta que agrupa um conjunto de informações. Ex: Echy possuí nome, idade, gênero e pertence à classe pessoa.

Atributo: Elementos que definem estrutura de uma classe, são chamados também de variáveis de classe e podem ser divididos em dois tipos básicos: atributos de instância e de classe. Os valores dos atributos de instância determinam o estado de cada objeto.

Método: Procedimento que atribui responsabilidades ao objeto, o que ela vai executar no algoritmo.

Modificador de acesso: Também chamado de visão de método ou visão de atributo, é a palavra-chave que define um atributo, método ou classe como público, privado ou protegido. Public o deixa aberto a modificações externas do usuário de forma mais simples, enquanto o Private o deixa privado e passa a ser necessário um método de acesso para poder modificar o valor da variável dentro da class;

Método de acesso: Dão acesso a entidades externas para o usuário acessar informações do objeto. Dentro deles podemos fazer validações especificas!

Método Construtor: Sempre nomeado com mesmo nome da classe, sendo sempre chamado quando a própria classe é instanciada.

Herança: Já mencionada acima, permite criar uma nova classe a partir de uma já existente. Também chamadas de subclasses. Pode ser identificada quando percebemos que “há um tipo de” alguma coisa. Ex: Vendedor é um tipo de funcionário; Então: Vendedor extends Funcionário;

Passo a passo para criar um objeto funcional:

1 – Crie o arquivo que hospedará o objeto com seu nome;

2 – Defina seus atributos globais;

3 – Defina quais serão seus tipos de métodos (Acessores, construtores, etc)

7a