**LINGUAGENS DE PROGRAMAÇÃO**

* 1949 – primeira linguagem de programação (linguagem de máquina)
* Anos 50 – COBOL
* Entre 60 e 70 – paradigmas: C
* Década de 90 – Java, C#, JavaScript, Ruby

**Problemas computacionais:**

Objeto de discussão que possui instruções passo a passo que são mais facilmente resolvíveis em ambiente computacional

**Problemas de decisão** – PROBLEMA DECIDÍVEL

* Caracter lógico – sim ou não
* Ideia – pertencimento

Ex: “dado um número n inteiro positivo, determine se n é primo”

**Problemas de busca**

* Objetivo
  + Semelhante ao nome
  + X está em A?
* Recorrente em teoria de grafos
  + **Ex: clique**

**Problemas de otimização**

Objetivo:

* Maximizar, ou
* Minimizar uma função

“Linguagem de programação é o método padronizado composto por um conjunto de regras sintáticas e semânticas de implementação de um **código fonte**”.

**Conjunto de palavras e regras**

**Código fonte:**

* Tradução
* Interpretação

**Como um computador entende o programa?**

“Um programa é um amontoado de palavras se não for possível que o computador entenda” – **Processo de tradução**

Linguagem de alto nível (**programa fonte)** > compilador (vai executar a analise do programa) > traduzir numa linguagem de máquina (assembly) **(programa objeto)**

**Tradução** – execução mais rápida / programas menores. Ex: Java, C++

1. Geração do programa objeto
2. Execução do programa objeto

**Interpretação** - maior flexibilidade. Ex: JavaScript, Ruby

* Programa fonte executado diretamente

**Características de um programa**

**Desenvolvimento de programas:**

* Legibilidade
* Redigibilidade
* Confiabilidade
* Custo

**Legibilidade:** “estado que é legível”

* Facilidade de leitura
* Compreensão
* Ortogonalidade (coerência nas instruções)
* Definição adequada das estruturas

**Redigibilidade:**  facilidade de escrita de código

* Pode conflitar com a legibilidade
* Ortogonalidade
* Simplicidade da escrita (coerência nas instruções)
* Suporte à abstração (separação)
* Reuso do código
* Expressividade

**Confiabilidade**: o código faz o que foi programado para fazer

* Verificação de tipos
* Trata de exceções
* Uso de ponteiros
* Compatibilidade entre compiladores

**Custo**: análise de impacto

Possui:

* Treinamento
* Codificação
* Compilação
* Execução
* Infra-estrutura

**Outras características:**

* Atualizações
* Uso para IA
* Disponibilidade de ferramentas
* Comunidade ativa
* Adoção pelo mercado

**Análise de código**

**Análise Léxica** - primeira fase de um processo de compilação e sua função é fazer a leitura do programa fonte, caractere por caractere e agrupá-los

* **Particionar** – identificar elementos denominados tokens
* **Classificar** – elementos: identificadores, palavras reservadas, números, strings, etc
* **Eliminar** – elementos: caracteres de espaços em branco, comentários, etc

**Análise Sintática** – define a corretude do programa

Sintaxe (forma):

“Componente do sistema linguístico que interligam os constituintes da sentença, atribuindo-lhe uma estrutura”

Padrão – gramática

* Depende da linguagem de programação utilizada

**Análise semântica** – trata da lógica do programa

Semântica (significado):

“É o estudo do significado. Incide sobre a relação entre significantes, como: palavras, frases, sinais e símbolos”

Erro de semântica:

* Não faz o que é esperado

Ex:

*if (x* ***=*** *0)*

*printf(“O valor e’ nulo\n”);* **Erro de semântica**

*if (x* ***==*** *0)*

*printf(“O valor e’ nulo\n”);* **Correto**

**Paradigmas de programação**

**Definição formal:**

“forma de resolução de problemas com diretrizes e limitações específicas de cada paradigma, utilizando linguagem de programação”

Possui regras para resolução de problema e está limitado à diretrizes específicas

**Classificação:**

* **Orientação à objeto**
* Procedural – chamadas sucessivas e procedimentos separados
* Funcional – instruções são baseadas em funções
* **Estruturado** – estrutura de blocos aninhados
* Computação distribuída – funções executadas de forma independente
* Lógico

**Paradigma Estruturado –** linguagem C

Conceitos:

* **Sequência**
* Decisão – teste lógico
* Iteração – funções, laços, condições. Dominou o mercado até POO (programação orientada à objeto)

Utilização:

* Problemas simples e diretos
* Aprender programação

POO ainda não compreendida por muitos

Ex:

function fatorial (x) {

if (x > 1) {

return x\*fatorial (x-1);

}

}

**Orientação à Objeto**

Paradigma de programação baseado na utilização de objetos e suas interações

Análogo ao mundo real

O que é um objeto? Violão, robô, bota, panela, carro, etc.

Dentro do mundo de POO: mulher, gato, bolo, montanha

“Um objeto é descrito por **características específicas, comportamentos e estados**”

* **Característica (atributos)**: o que eu tenho
* **Comportamento (métodos)**: o que sou capaz de fazer
* **Estado**: como eu faço

Ex: **Caneta**

* **O que eu tenho**: Modelo, Cor, Carga, Corpo, Tampa, Ponta
* **O que sou faz de fazer**: escrever, desenhar, rabiscar, pintar, destampar
* **Como eu faço**: Tampada, Destampada, Em uso

**Ponto de vista da programação. O que é um objeto?**

**POO:**

**Classe**

* Alocação em memória
* Operações associadas

**Estruturada**

* Alocação em memória

Variável

* Operações associadas

Ex:

classe caneta():

cor: inteiro

carga: caractere

tampada: lógico

método escrever ()

Se estado tampada

Escreva (“Destampar”)

Senão

Escreva (“algo”)

Fimse

Método tampar()

Tampada = verdadeiro

fimclasse

**Pilares de Orientação a Objeto**

* Herança
* Encapsulamento
* Polimorfismo
* Abstração

**Herança** – uma classe filha herda as características de uma classe mãe. Atributos, métodos, comportamentos, mas pode ter as características dela.

* Classes
* Características
  + Atributos
  + Métodos

**POO** – reuso de código

**PE** (paradigma estruturado) – problemas específicos e diretos