bParadigma na Programação

* + O Paradigma é a Base da Solução
    - O Paradigma é a Base da Solução

Diagrama, Carta

Descrição gerada automaticamente

* + Paradigma Imperativo (Década de 50)
    - Contexto
      * O paradigma imperativo foi um dos primeiros estilos de programação adotados, em uma época em que o foco era controlar diretamente os recursos do computador
    - Linguagens iniciais
      * Assembly, Fortran
    - Características
      * Comandos sequenciais, uso de variáveis para armazenar estado e controle explícito do fluxo de execução.
    - Impacto
      * Esse paradigma serviu como a base para a programação estruturada e oferece controle direto sobre o hardware.
  + Paradigma Funcional (Década de 1950)
    - Contexto
      * Baseado na teoria matemática das funções, esse paradigma cresceu com a popularização de linguagens específicas
    - Linguagens
      * Lisp, Haskell.
    - Características
      * O paradigma funcional evita estados mutáveis e efeitos colaterais, priorizando funções puras e imutabilidade.
    - Impacto
      * Oferece vantagens em programação concorrente e processamento de dados, sendo muito utilizado em problemas de matemática e computação científica, assim como em frameworks modernos para manipulação de dados e concorrência.
  + Paradigma Procedural (Década de 1960)
    - Contexto
      * paradigma procedural surgiu como uma evolução natural do imperativo, ao introduzir estruturas que organizam o código em procedimentos ou sub-rotinas. Isso ajudou a tornar o código mais modular e reutilizável.
    - Linguagens
      * C, Fortran, COBOL, Pascal
    - Características
      * Organização do código em procedimentos ou funções que podem ser chamados para realizar tarefas específicas. O programa é construído como uma sequência de chamadas a esses procedimentos, o que facilita o fluxo controlado de execução
    - Vantagens
      * Melhor modularidade em comparação ao imperativo puro, permitindo a reutilização de código e a redução de redundância.
    - Exemplos de uso:
      * Aplicações científicas, engenharia e sistemas empresariais que requerem processos repetitivos e organizados

Diagrama

Descrição gerada automaticamente

* + Paradigma Estruturado (Década de 60)
    - Contexto
      * Conforme os programas se tornaram mais complexos, surgiu a necessidade de criar uma estrutura mais organizada para o código.
    - Linguagens
      * C, Pascal.
    - Características
      * Introdução de estruturas de controle (como loops e condicionais), subrotinas e funções, facilitando a manutenção e legibilidade do código
    - Impacto
      * O paradigma estruturado abriu caminho para o desenvolvimento de programas mais organizados, reduzindo a "codificação spaghetti" (código desorganizado e de difícil leitura).

Diagrama

Descrição gerada automaticamente

* + Paradigma Lógico (Década de 1970 e 1980)
    - Contexto
      * Criado para aplicações de inteligência artificial e resolução de problemas baseados em lógica
    - Linguagens
      * Prolog, Datalog.
    - Características
      * Foca em declarar relações e regras que descrevem o problema, deixando para o computador a busca de soluções lógicas
    - Impacto
      * Embora não seja amplamente utilizado para aplicações comerciais, é essencial em áreas de IA, como linguagens naturais e inferência lógica
  + Paradigma Orientado a Objetos (OO) (Década de 80)
    - Contexto
      * Surgiu como resposta à necessidade de modularidade e reutilização de código para sistemas cada vez mais complexos
    - Linguagens
      * Java, C++.
    - Características
      * Organiza o código em "objetos" que representam entidades do mundo real ou conceitual, encapsulando dados e comportamentos.
    - Impacto
      * Tornou-se o paradigma dominante em desenvolvimento corporativo e jogos, promovendo a criação de bibliotecas reutilizáveis e ajudando a gerenciar a complexidade de sistemas grandes.

Texto

Descrição gerada automaticamente

* + Paradigma Reativo (2000 em diante)
    - Contexto
      * Com o aumento da demanda por sistemas em tempo real e responsivos, o paradigma reativo se tornou mais relevante
    - Ferramentas e Tecnologias:
      * RxJS, Reactor, frameworks de programação reativa.
    - Características
      * Utiliza fluxos de dados assíncronos e programação orientada a eventos, onde mudanças em uma parte do sistema reagem automaticamente a mudanças em outra.
    - Impacto
      * Essencial para aplicações modernas como dashboards em tempo real, plataformas de streaming e sistemas distribuídos.

Linha do tempo

Descrição gerada automaticamente com confiança baixa

* + Paradigmas Concorrentes e Paralelos (ao longo das décadas)
    - Contexto
      * Com a chegada dos processadores multicore, a programação concorrente e paralela tornou-se necessária para aproveitar melhor o hardware.
    - Ferramentas e Tecnologias:,
      * Threads, OpenMP, MPI, frameworks de atores como Akka.
    - Características
      * : Foco na execução simultânea de múltiplas tarefas para aumentar o desempenho e a eficiência de aplicativos que processam grandes volumes de dados.
    - Impacto
      * Essencial para áreas como computação científica, simulações, e big data, ajudando a criar programas mais rápidos e capazes de processar grandes volumes de dados em paralelo.

Diagrama

Descrição gerada automaticamente

* + Paradigma Imperativo e Declarativo

Diagrama

Descrição gerada automaticamente

Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente

* + Preparando uma xícara de café usando os paradigmas
    - Imperativo
      * "João, pegue a cafeteira. Coloque água até o nível indicado. Abra o compartimento do pó de café. Coloque três colheres de pó. Feche o compartimento. Coloque a cafeteira na base. Aperte o botão de ligar. Aguarde até o café ficar pronto. Pegue uma xícara e coloque o café nela."
    - Declarativo
      * "João, faça uma xícara de café."