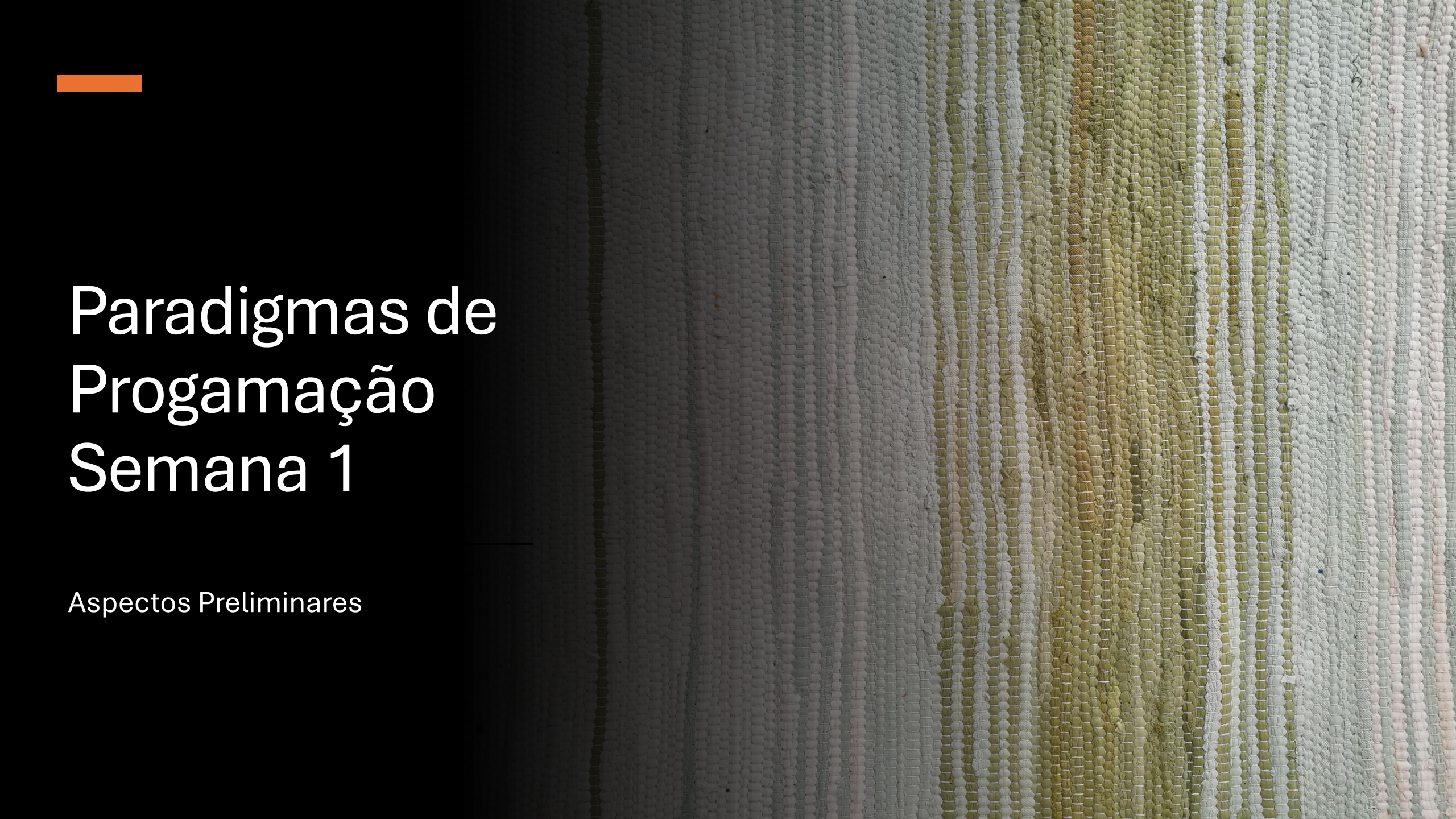
Paradigmas de Progamação Semana 1

Aspectos Preliminares



RAZÕES PARA ESTUDAR CONCEITOS DE LINGUAGENS DE PROGRAMAÇÃO

- Estudar conceitos de linguagens de programação não só oferece benefícios técnicos, como também promove o desenvolvimento de habilidades transferíveis e abre portas para uma variedade de oportunidades profissionais.

Nº	Razões para Estudar Conceitos de Linguagens de Programação
1	Desenvolver habilidades analíticas e de resolução de problemas
2	Entender os fundamentos da computação e da lógica de programação
3	Capacidade de expressar soluções para problemas de forma estruturada e precisa
4	Melhorar a capacidade de comunicação e colaboração com outros desenvolvedores
5	Explorar diferentes paradigmas de programação e suas aplicações
6	Adquirir conhecimentos sobre algoritmos e estruturas de dados
7	Desenvolver aplicações eficientes e otimizadas
8	Abrir portas para oportunidades de carreira em tecnologia da informação
9	Fornecer uma base sólida para aprender novas linguagens de programação no futuro
10	Capacidade de criar soluções personalizadas para problemas específicos em diversas áreas

DOMÍNIOS DE PROGRAMAÇÃO

Desenvolvimento de Software: Criação de aplicativos, sistemas e programas de computador para diversos propósitos, como aplicativos móveis, web e sistemas embarcados. Linguagens como Python, Java, JavaScript e C++ são usadas para projetar, codificar e testar soluções de software.

Ciência de Dados e Análise: Coleta, análise e interpretação de grandes conjuntos de dados para obter insights. Programadores usam linguagens como Python, R e SQL para limpar, processar e visualizar dados, aplicando técnicas de aprendizado de máquina e inteligência artificial.

Desenvolvimento Web: Criação de sites interativos, aplicativos da web e serviços online. Utiliza linguagens como HTML, CSS e JavaScript, juntamente com frameworks como React, Angular e Django, para acelerar o desenvolvimento e criar experiências digitais dinâmicas.

Automação e Automação de Processos: Otimização de processos e tarefas repetitivas em diversos setores. Programadores escrevem scripts e programas para automatizar fluxos de trabalho, utilizando linguagens como Python, PowerShell e Bash.

Jogos e Entretenimento Digital: Desenvolvimento de jogos eletrônicos, simulações interativas e experiências imersivas. Utiliza engines de jogos como Unity e Unreal Engine, juntamente com linguagens como C#, C++ e JavaScript, para criar experiências envolventes aos jogadores.

Critérios de avaliação de linguagens

Legibilidade: Este critério refere-se à facilidade de compreender o código fonte escrito em uma determinada linguagem. Uma linguagem com uma sintaxe clara e intuitiva facilita a leitura e compreensão do código, tanto para os programadores que o escrevem quanto para aqueles que precisam mantê-lo posteriormente. Exemplo: Python é frequentemente elogiado por sua legibilidade devido à sua sintaxe limpa e semântica clara, o que facilita a compreensão do código.

Facilidade de Escrita: Este critério considera o quão fácil é escrever código em uma linguagem específica. Uma linguagem que permite aos programadores expressar suas ideias de forma rápida e concisa pode aumentar a produtividade e reduzir a probabilidade de erros. Exemplo: Ruby é conhecido por sua sintaxe concisa e expressiva, que permite aos desenvolvedores escrever código de maneira rápida e eficiente.

Confiabilidade: A confiabilidade de uma linguagem de programação refere-se à consistência e previsibilidade do comportamento do código escrito nessa linguagem. Isso inclui a capacidade da linguagem de detectar e lidar com erros de forma eficaz, bem como a estabilidade do ambiente de execução. Exemplo: Rust é elogiado por sua forte segurança de memória e sistema de tipos que previne muitos tipos de erros comuns de programação, tornando-o uma escolha confiável para sistemas críticos.

Critérios de avaliação de linguagens e as características que os afetam

Critérios de avaliação de linguagens e as características que os afetam

	CRITÉRIOS		
Característica	**LEGIBILIDADE**	**FACILIDADE DE ESCRITA**	**CONFIABILIDADE**
Simplicidade	•	•	•
Ortogonalidade	•	•	•
Tipos de dados	•	•	•
Projeto de sintaxe	•	•	•
Suporte para abstração		•	•
Expressividade		•	•
Verificação de tipos			•
Tratamento de exceções			•
Apelidos restritos			•

[1] O quarto critério principal é o custo, que não foi incluído na tabela porque é apenas superficialmente relacionado aos outros três critérios e às características que os influenciam.

Influências no projeto de linguagens

- A arquitetura de computadores tem uma influência substancial no projeto de linguagens de programação, afetando diversos aspectos, incluindo:

1. **Modelo de Memória**:
    1. **Fatores Influenciados**: A maneira como a arquitetura de computadores organiza e acessa a memória afeta as decisões de projeto relacionadas ao gerenciamento de memória das linguagens de programação.
    2. **Exemplo**: Linguagens como C e C++ foram projetadas com um modelo de memória próximo ao hardware, permitindo manipulação direta de endereços de memória. Em contraste, linguagens como Java e Python utilizam um modelo de memória gerenciado, onde o sistema de tempo de execução lida com alocação e desalocação de memória.
2. **Arquitetura de Conjunto de Instruções (ISA)**:
    1. **Fatores Influenciados**: A arquitetura de conjunto de instruções de um processador influencia a forma como as linguagens de programação são compiladas e otimizadas.
    2. **Exemplo**: Linguagens como Assembly estão intimamente ligadas à arquitetura de conjunto de instruções de um processador específico, permitindo controle direto sobre o hardware. Linguagens de alto nível como C são compiladas para o código de máquina específico do conjunto de instruções da arquitetura alvo, possibilitando otimizações de desempenho específicas do hardware.
3. **Desempenho e Eficiência**:
    1. **Fatores Influenciados**: A capacidade de uma linguagem de programação de aproveitar eficientemente os recursos de hardware está diretamente ligada à arquitetura de computadores.
    2. **Exemplo**: Linguagens como C e C++ permitem controle granular sobre o uso de recursos de hardware, possibilitando otimizações de desempenho de baixo nível. Isso as torna adequadas para o desenvolvimento de sistemas de tempo real e aplicativos de alto desempenho.
4. **Portabilidade**:
    1. **Fatores Influenciados**: A portabilidade de uma linguagem de programação é influenciada pela capacidade de executar o código em diferentes arquiteturas de computadores.
    2. **Exemplo**: Linguagens como Java são projetadas para serem altamente portáveis, executando em uma máquina virtual Java (JVM) que pode ser implementada em uma variedade de arquiteturas de computadores. Isso permite que o mesmo código Java seja executado em diferentes sistemas sem a necessidade de alterações significativas.

CATEGORIAS DE LINGUAGENS

Paradigma Imperativo/Procedural:

Características: Baseia-se em uma sequência de instruções que alteram o estado do programa.

Exemplos: C, Pascal, BASIC.

Paradigma Orientado a Objetos (OO):

Características: Organiza o código em objetos que podem conter dados e métodos.

Exemplos: Java, C++, Python, C#.

Paradigma Funcional:

Características: Baseia-se no conceito de funções matemáticas puras e evita o estado compartilhado e mutável.

Exemplos: Haskell, Lisp, Clojure, Erlang.

Paradigma Lógico:

Características: Baseia-se na lógica formal e na resolução de problemas através de inferência lógica.

Exemplos: Prolog, Datalog, Mercury.

TRADE-OFFS NO PROJETO DE LINGUAGENS

Os critérios de avaliação de linguagens de programação fornecem um framework para o projeto, porém são frequentemente contraditórios.

Segundo Hoare (1973), a reconciliação desses critérios é uma das principais tarefas de engenharia.

Um exemplo de conflito é entre confiabilidade e custo de execução, exemplificado pela diferença entre Java e C na verificação de índices de vetores.

Outro exemplo é a trade-off entre facilidade de escrita e legibilidade, ilustrada pelo caso da linguagem APL, que é expressiva, mas pouco legível.

O projeto de linguagens frequentemente envolve comprometimentos e trade-offs devido aos conflitos entre os critérios de avaliação.

MÉTODOS DE IMPLEMENTAÇÃO

Compilação: Transforma o código-fonte em código de máquina diretamente executável pelo hardware. Exemplos: C, C++, Java (para bytecode) e Rust.

Interpretação: Executa o código linha por linha por um interpretador, resultando em execução mais lenta. Exemplos: Python, JavaScript e Ruby.

Compilação Just-in-Time (JIT): Combina compilação e interpretação, otimizando partes do código durante a execução. Exemplos: Java (JVM) e C# (CLR).

Pré-processamento: Manipula o código-fonte antes da compilação ou interpretação. Exemplos: Pré-processamento em C/C++ para diretivas como #include e #define.

Máquinas Virtuais: Executam o código em uma máquina virtual específica. Exemplos: JVM para Java, CLR para C#.