

# Linguagens de Programação

---

## Introdução

Professor Thiago Alves

# Informações Gerais

---

- Aulas Teóricas
- Avaliação
  - Provas N1
  - Provas N2
  - Seminários
- Frequência
- Aulas de Exercícios
  - Pontos de participação

# Informações Gerais

---

- Bibliografia

- **Conceitos de Linguagens de Programação - Robert W. Sebesta**



# Tópicos

---

- Motivação
- Domínio de programação
- Critérios para avaliar linguagens
- Influência na construção de uma linguagem
- Categorias de Linguagens
- Trade-Offs da construção de linguagens

# Motivação

---

- Aumentar a habilidade de expressar ideias
- Melhor background para escolher linguagens apropriadas
- Facilitar o aprendizado de linguagens
- Melhor utilização das linguagens
- Avanços gerais da Computação

# Domínios de Programação

---

- Aplicações científicas
  - Números grandes em ponto flutuante; Uso de arrays
  - Fortran
- Aplicações de negócios
  - Produzir relatórios, Usar números decimais e caracteres
  - COBOL
- Inteligência artificial
  - Manipulação de símbolos no lugar de números; Uso de listas encadeadas.
- Programação de sistemas
  - Necessita de eficiência pelo uso contínuo
  - C
- Desenvolvimento Web
  - Coleção de linguagens: marcação (XHTML), scripting (PHP), propósito geral (Java)
- Desenvolvimento para Dispositivos Móveis: C#, Kotlin, Swift, Java

# Critérios de Avaliação

---

- **Legibilidade:** a facilidade com que os programas podem ser lidos e entendidos
- **Writability (Escrita):** a facilidade com que uma linguagem pode ser usada para criar programas
- **Confiança:** realização das suas especificações
- **Custo:** o custo total final

# Legibilidade

---

- Simplicidade global
  - Conjunto suficiente de construtores básicos
  - Características Múltiplas Mínimas: poucas formas de realizar uma mesma operação
- Incremento
  - Overloading de operadores mínimo
- Operador +
  - Simplicidade extrema também é problema
- Ortogonalidade
  - Um conjunto relativamente pequeno de primitivas podem ser combinadas em um número relativamente pequeno de maneiras
- Tipos de dados, ponteiros, vetores



# Legibilidade

---

- Tipos de dados
  - Tipo de dados predefinidos adequados
- Linguagens sem boolean
- Considerações de sintaxe
  - Palavras especiais e formas de compor comandos
- Início e fim de blocos: {, }, endif
- Conflito entre legibilidade e simplicidade
  - Forma e significado: construtores auto-explicativos

# Writability (Escrita)

---

- Simplicidade e ortogonalidade
  - Poucos construtores, número pequeno de primitivas, conjunto pequeno de regras de combinação
- Suporte para abstração
  - A habilidade de definir e usar estruturas ou operações complexas de forma que permita ignorar detalhes
  - Ex: funções, métodos
- Expressividade
  - Um conjunto de maneiras relativamente convenientes de especificar certas operações
- Incremento, for

# Confiança

---

- Checagem de tipos

- Teste para erros de tipo
- Parâmetros de versões antigas do C

- Tratamento de exceções

- Interceptar erros de execução e tomar medidas de correção
- C++ e Java possuem

- Aliasing

- Dois ou mais nomes distintos para acessar uma mesma célula de memória

- Legibilidade e writability (escrita)

- Uma linguagem que não suporta formas naturais de expressar um algoritmo necessitará de abordagens não naturais. Isso reduz a confiança

# Custo

---

- Treinar os programadores
  - Depende da simplicidade e ortogonalidade
- Escrever os programas
  - Depende da writability e da aplicação
  - Custo diminuído com o uso de ambientes de desenvolvimento
- Compilar programas
- Executar programas
- Sistema de implementação da Linguagem: disponibilidade de compiladores grátis.

# Custo

---

- Confiança: pouca confiança leva a altos custos
  - Sistemas críticos: usina nuclear, máquina de raio-X, naves espaciais, sistemas financeiros
  - Falhas acarretam em enormes custos
- Manutenção de programas
  - Correções
  - Incremento de novas funcionalidades
  - Programados diferentes dos originais
  - Depende muito da legibilidade e writability

# Outros Critérios de Avaliação

---

- Portabilidade
  - A facilidade com que programas podem ser movidos de uma implementação para outra
- Generalidade
  - A aplicação em uma vasta gama de aplicações
- Well-definedness (Bem definida)
  - Definição oficial da linguagem ser completa e precisa

# Influências no Projeto da Linguagem

---

- Arquitetura do Computador
  - Linguagens são desenvolvidas considerando a prevalente arquitetura *von Neumann*
- Metodologias de Programação
  - Novas metodologias de desenvolvimento (ex: orientação a objetos) levaram a novos paradigmas de programação
  - Novas linguagens de programação

# Influência da Arquitetura

---

- Linguagens Imperativas são as mais dominantes por conta da arquitetura de von Neumann
  - Dados e programas armazenados na memória
  - Memória separada da CPU
  - Instruções e dados são trazidos da memória para a CPU
  - Base para linguagens imperativas
- Variáveis são modelos de células de memória
- Atribuição modela o fluxo da memória para a CPU
- Iteração é eficiente pois as instruções são armazenadas sequencialmente na memória
- Repetição de uma seção de código requer apenas um desvio na sequência



# Influências de Metodologias

---

- 1950 e começo de 1960: aplicações simples; preocupação com eficiência da máquina
- Final de 1960: eficiência das pessoas se tornou importante; legibilidade, estruturas de controle
  - Programação estruturada
  - Projeto top-down e refinamento passo a passo
- Final de 1970: processos orientados aos dados
  - Abstração de dados
  - Modelagem de dados e tipos abstratos de dados
- Meio de 1980: Orientação a Objetos
  - Abstração de dados + herança + polimorfismo

# Categorias de Linguagens

---

- Imperativa
  - Variáveis, atribuição, e iteração
  - Inclui linguagens que suportam OO
  - Inclui linguagens de script
  - Inclui linguagens visuais
  - Exemplos: C, Java, Perl, JavaScript, Visual BASIC, .NET, C++
- Lógica
  - Baseada em regras (regras são especificadas em qualquer ordem)
  - Exemplo: Prolog
- Marcação/Híbrida
  - Exemplos de Marcação: XHTML, XML
  - Linguagens de marcação estendidas para suportar programação

# Trade-Offs

---

- **Confiança vs. Custo de execução**

- Java faz uma checagem da indexação de arrays. Isto acarreta um aumento no custo de execução
- C não faz isso

- **Legibilidade vs. Writability**

- APL fornece vários operadores poderosos (vários símbolos), permitindo computações complexas escritas em um programa compacto
- Custo de uma pobre legibilidade

- **Writability (flexibilidade) vs. confiança**

- Ponteiros em C++ são poderosos e flexíveis mas devem ser usados com cuidado

# Métodos de Implementação

---

- Compilação

- Programas são traduzidos em instruções de máquina
- Depois de compilado basta utilizar o executável
- Gasta um tempo maior na compilação

- Interpretação

- Programas são executados a medida que o interpretador percorre o programa
- O processo de interpretação é mais rápido
- Necessita ser interpretado em toda execução

# Métodos de Implementação

---

- Híbrido
  - Meio termo entre compilação e interpretação pura
  - Mais rápido que a interpretação pura

# Compilação

---

- Traduz programas em alto-nível (linguagem fonte) em código de máquina
- Tradução lenta
- Execução rápida
- Várias fases:
  - Análise sintática
  - Análise Semântica: gera código intermediário
  - Geração de código: código de máquina é gerado

# Gargalo deVon Neumann

---

- Velocidade de conexão entre memória e CPU determina a velocidade do computador
- Instruções de programas geralmente são executados mais rapidamente que a velocidade da conexão
- A velocidade da conexão é um gargalo
- É um fator limitante primário na velocidade dos computadores

# Interpretação Pura

---

- Sem tradução
- Implementação mais fácil
- Erros em tempo de execução são mais específicos
  - Índice de vetor fora da faixa
- Execução mais lenta
- Geralmente requer mais espaço



# Implementação Híbrida

---

- Traduzem programas em alto nível para uma linguagem intermediária
- Interpreta a linguagem intermediária
- Pearl: parcialmente compilado para detectar erros antes da interpretação
- Java: traduzido para bytecode e interpretado
  - Portabilidade para máquinas com interpretador bytecode (JVM)

# Implementação Just-in-Time (JIT)

---

- Inicialmente traduz programas em uma linguagem intermediária
- Compila a linguagem intermediária de subprogramas em código de máquina quando são chamados
- Versão de código de máquina é mantida para chamadas posteriores
- Sistemas JIT são vastamente usados para programas em JAVA
- Linguagem .NET é implementada com sistemas JIT

# Pré-processadores

---

- Macros (instruções) são usados para especificar que o código de outro arquivo deve ser incluído
- Um pré-processador processa um programa imediatamente antes do programa ser compilado
- Um exemplo bem conhecido: pré-processador de C
  - `#include`, `#define` e macros similares

# Resumo

---

- O estudo de linguagens de programação é importante:
  - Aumenta a capacidade de usar diferentes construtores
  - Permite escolher linguagem de forma mais inteligente
  - Aprender novas linguagens é fica mais fácil
- Critérios importantes para avaliar linguagens:
  - Legibilidade, writability, confiança, custo
- Maiores influências no projeto de linguagens tem sido a arquitetura das máquinas e metodologias de desenvolvimento de software
- Métodos de implementação de linguagens: compilação, interpretação pura e híbrida