PARADIGMAS E LINGUAGENS DE PROGRAMAÇÃO

ENGENHARIA DA COMPUTAÇÃO – UFC/SOBRAL

Prof. Danilo Alves

danilo.alves@alu.ufc.br

MOTIVAÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- Linguagens na Engenharia da computação;
- Cientista e programador;
- Princípios e fundamentos das linguagens;
- Poliglota iletrado e linguista teórico;

OBJETIVOS

UNIVERSIDADE FEDERAL DO CEARÁ

- Entendimento dos fundamentos das linguagens;
- Compreender, identificar e descrever características dos diversos paradigmas;
- Implementar linguagens que utilizem os paradigmas abordados;

CONTEÚDO PROGRAMÁTICO DA DISCIPLINA

UNIVERSIDADE FEDERAL DO CEARÁ

- Visão geral e evolução das principais linguagens e paradigmas;
- Descrição de Sintaxe e Semântica;
- Valores e tipos;
- Variáveis, vinculações e verificações de tipos;
- Expressões e Instruções de atribuição;
- Abstrações;

CONTEÚDO PROGRAMÁTICO DA DISCIPLINA

UNIVERSIDADE FEDERAL DO CEARÁ

- Estudo sobre paradigma funcional;
- Implementação de linguagem funcional;
- Estudo sobre o Paradigma Lógico;
- Implementação de linguagem lógica;
- Estudo sobre o Paradigma Orientado a objetos;
- Implementação de linguagem orientada a objetos;

AVALIAÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- 3 avaliações:
  - 2 provas;
  - 1 projeto;
- As provas podem apresentar aspectos teóricos;
- Listas de problemas ao longo da disciplina;
- Atividades para arredondamento
  - Nota>=6.9
  - Nota>=3.9

BIBLIOGRAFIA

UNIVERSIDADE FEDERAL DO CEARÁ

MELO, A. C. V. de; SILVA, F. S. C da. "Princípios de Linguagens de Programação", 1ª Edição, São Paulo: Edgard Blücher, 2003.

SEBESTA, R. W. "Conceitos de linguagens de programação", 5ª Edição, Porto Alegre: Bookman, 2003.

WATT, D.; FINDLAY, W. "Programming Language Design Concepts", 1ª Edição, John Wiley & Sons, 2004.

ASPECTOS PRELIMINARES DE LINGUAGENS DE PROGRAMAÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

PARADIGMAS DE PROGRAMAÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- Imperativa
  - Características centrais são variáveis, sentenças de atribuição e de iteração
  - Inclui linguagens que suportam programação orientada a objeto
  - Inclui linguagens de *scripting*
  - Inclui as linguagens visuais
  - Exemplos: C, Java, Perl, JavaScript, Visual BASIC .NET, C++
- Funcional
  - Principais meios de fazer os cálculos é pela aplicação de funções para determinados parâmetros
  - Exemplos: LISP, Scheme

PARADIGMAS DE PROGRAMAÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- Lógica
  - Baseada em regras (regras são especificadas sem uma ordem em particular)
  - Exemplo: Prolog

- Orientada a Objetos
  - Utiliza representação do mundo real através de modelos
  - Visão próxima dos humanos
  - Exemplo: Java, C++, PHP 5

RAZÕES PARA ESTUDAR CONCEITOS DE LINGUAGENS DE PROGRAMAÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- Capacidade aumentada para expressar ideias
- Embasamento melhorado para escolher linguagens apropriadas
- Melhor entendimento da importância da implementação
- Habilidade aumentada para aprender novas linguagens
- Avanço geral da computação

RAZÕES PARA ESTUDAR CONCEITOS DE LINGUAGENS DE PROGRAMAÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- Capacidade aumentada para expressar ideias
  - Capacidade intelectual pode ser influenciada pelo poder expressivo da linguagem
  - Uma maior compreensão de uma LP pode aumentar nossa habilidade em pensar em como atacar os problemas.
  - Conhecimento amplo dos recursos de linguagem reduz as limitações no desenvolvimento de softwares
  - A melhor compreensão das funções e implementação das estruturas de uma LP nos leva a usar a LP de modo a extrair o máximo de sua funcionalidade e eficiência
  - Recursos ou facilidades podem ser simulados

RAZÕES PARA ESTUDAR CONCEITOS DE LINGUAGENS DE PROGRAMAÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- Embasamento melhorado para escolher linguagens apropriadas
  - Escolher a melhor linguagem para um problema específico devido ao conhecimento de novos recursos é difícil para:
    - Programadores antigos
    - Desenvolvedores sem educação formal

RAZÕES PARA ESTUDAR CONCEITOS DE LINGUAGENS DE PROGRAMAÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- Melhor entendimento da importância da implementação
  - Leva um entendimento do PORQUÊ das linguagens serem projetadas de determinada maneira.
  - Melhora as escolhas que podemos fazer entre as construções de LP e as consequências das opções.

RAZÕES PARA ESTUDAR CONCEITOS DE LINGUAGENS DE PROGRAMAÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- Habilidade aumentada para aprender novas linguagens
  - Aprendizado contínuo é fundamental, a computação é uma ciência nova
  - Compreender os conceitos gerais das linguagens torna mais fácil entender como eles são incorporados na linguagem que está sendo aprendida

RAZÕES PARA ESTUDAR CONCEITOS DE LINGUAGENS DE PROGRAMAÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- Avanço geral da computação
  - Nem sempre as linguagens mais populares são melhores, por quê?

    - Imposição!!
  - Por que existem várias linguagens de programação?

    - Resolução específica de problemas

DOMÍNIOS DE PROGRAMAÇÃO

EM QUE ÁREAS É REQUERIDA UMA VISÃO DIFERENCIADA DE UMA LINGUAGEM DE PROGRAMAÇÃO?

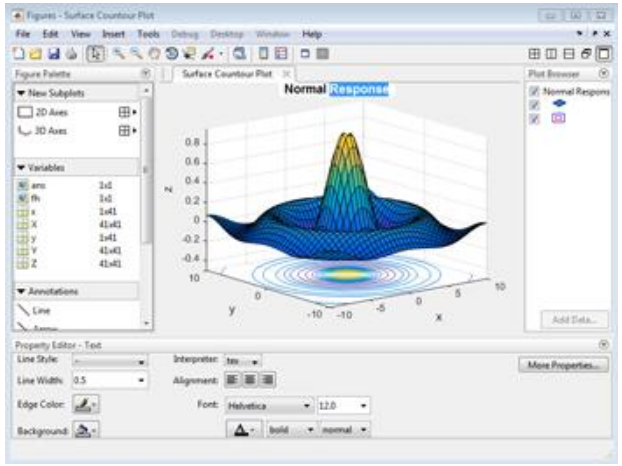
COMPUTADORES TÊM SIDO APLICADOS A UMA INFINIDADE DE ÁREAS;

UNIVERSIDADE FEDERAL DO CEARÁ

DOMÍNIOS DE PROGRAMAÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- Aplicações científicas
  - As aplicações científicas incentivaram a criação de algumas linguagens de alto nível, como por exemplo o FORTRAN;
  - O ALGOL 60 e a maioria de suas descendentes também se destinam a serem usadas nessa área, ainda que projetadas para outras áreas relacionadas;
  - MATLAB;



DOMÍNIOS DE PROGRAMAÇÃO

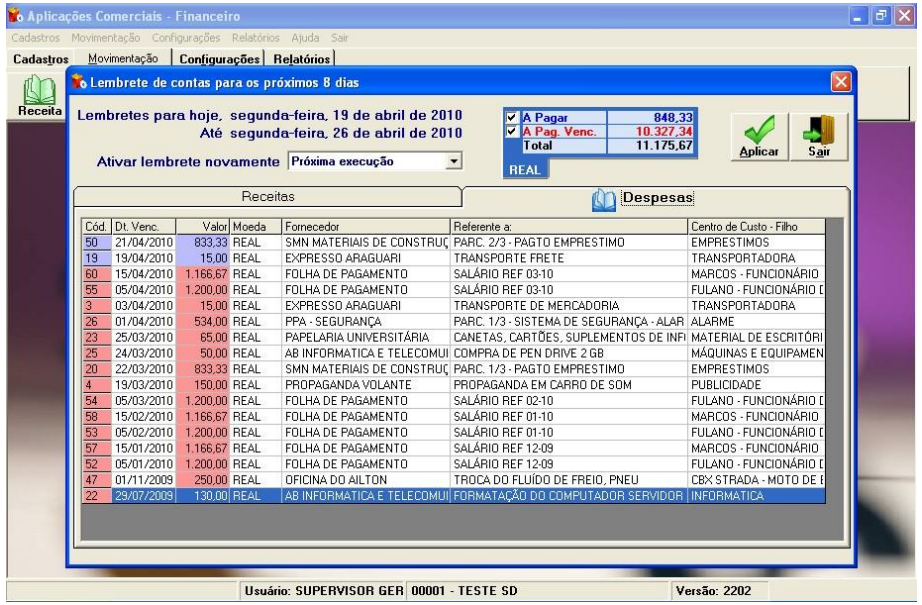
UNIVERSIDADE FEDERAL DO CEARÁ

- Aplicações comerciais
  - produção de relatórios elaborados;
  - A primeira linguagem bem sucedida foi o COBOL;

- Inteligência artificial
  - Computação de símbolos em vez de números;

processamento de dados em listas;

  - LISP e PROLOG



DOMÍNIOS DE PROGRAMAÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- Programação de sistemas
  - possuir eficiência na execução por propiciar suporte a execução de outros aplicativos;
  - permite ao software fazer interface com os dispositivos externos (Sistema Operacional);
  - C;

- Linguagens de scripts
  - Manipulação textual simples com interfaceamento de software já existente;
  - Muito utilizadas para criar páginas dinâmicas;
  - Perl, Python, PHP;

CRITÉRIO DE AVALIAÇÃO DE LINGUAGENS

UNIVERSIDADE FEDERAL DO CEARÁ

- Legibilidade: facilidade com a qual os programas podem ser lidos e entendidos
- Facilidade de escrita: facilidade com a qual uma linguagem pode ser usada para criar programas para um dado domínio
- Confiabilidade: conformidade com as especificações
- Custo: o custo total definitivo de uma linguagem

LEGIBILIDADE

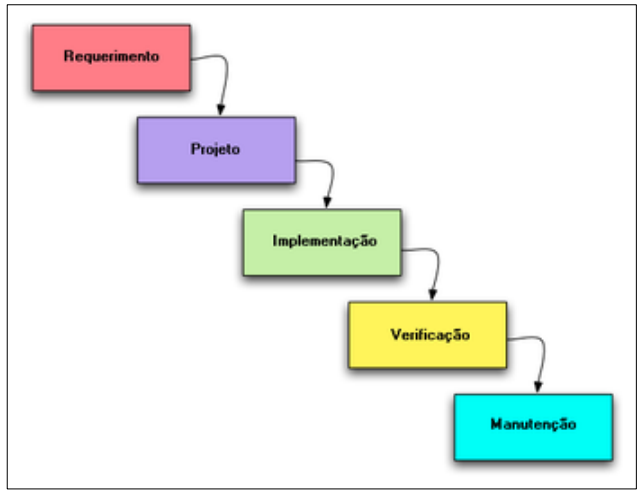
- Um dos critérios mais importantes para julgar uma LP é a facilidade com que os programas são lidos e entendidos
- Antes de 70: pensado em termos de escrita de código.

  - Principais características: eficiência e legibilidade de máquina.
  - LP foram projetadas mais do ponto de vista do computador do que

do usuário

- Na década de 70 foi desenvolvido o conceito de ciclo de vida de software

  - manutenção



"A facilidade de manutenção é determinada em grande parte pela legibilidade dos programas"

LEGIBILIDADE

UNIVERSIDADE FEDERAL DO CEARÁ

- Simplicidade geral

– Um conjunto controlável de recursos e construções

  - conjunto grande de recursos é mais difícil de aprender
  - Problema surge quando o programador aprende um recurso diferente daquele que está familiarizado

– Mínima multiplicidade de recursos

  - Ex.: há 4 formas de incremento de variável em C/C++/Java

– Mínima sobrecarga de operadores – operador possui mais de um significado

  - Útil, mas pode levar a problemas de legibilidade
  - Ex.: Operador + para somar vetores, números e concatenar

LEGIBILIDADE

- A simplicidade geral levada ao extremo prejudica a legibilidade.
  - Ex: a forma e significado da maioria das instruções em Assembly são modelos de simplicidade
- Falta instruções de controle mais complexas, torna necessário o uso de mais códigos para expressar problemas do que os necessário em linguagens de alto nível.

```
Endereço  Código        Assembly
--------- ------------- ------------------------------
1B8D:0100 01D8          ADD     AX,BX
1B8D:0102 C3            RET
1B8D:0103 16            PUSH    SS
1B8D:0104 B03A          MOV     AL,3A
1B8D:0106 380685D5      CMP     [D585],AL
1B8D:010A 750E          JNZ     011A
1B8D:010C 804E0402      OR      BYTE PTR [BP+04],02
1B8D:0110 BF86D5        MOV     DI,D586
1B8D:0113 C6460000      MOV     BYTE PTR [BP+00],00
1B8D:0117 E85F0B        CALL    0C79
1B8D:011A 8B7E34        MOV     DI,[BP+34]
1B8D:011D 007C1B        ADD     [SI+1B],BH
```

LEGIBILIDADE

UNIVERSIDADE FEDERAL DO CEARÁ

- Ortogonalidade

– Um conjunto relativamente pequeno de construções primitivas podem ser combinados em um número pequeno de maneiras para construir as estruturas de controle e de dados de uma linguagem

– Cada possível combinação é legal

- Ex: Tipos de dados primitivos para dados (inteiro, ponto flutuante, ponto flutuante de dupla precisão e caractere) e dois operadores de tipo (vetor e ponteiro).

  - Os operadores de tipo podem ser aplicados a eles mesmos e aos quatro tipos de dados primitivos;

"Quanto mais ortogonal é uma linguagem, menor é o número de exceções às regras da linguagem."

LEGIBILIDADE

UNIVERSIDADE FEDERAL DO CEARÁ

– Exemplo de falta de ortogonalidade em C

- Apesar de C ter duas formas de tipos de dados estruturados, vetores e registros (*structs*), os registros podem ser retornados por funções, mas os vetores não
- Um membro de uma estrutura pode ser de qualquer tipo de dados, exceto *void* ou uma estrutura do mesmo tipo
- Um elemento de um vetor pode ser qualquer tipo de dados, exceto *void* ou uma função
- Parâmetros são passados por valor, a menos que sejam vetores, o que faz com que sejam passados por referência

LEGIBILIDADE

UNIVERSIDADE FEDERAL DO CEARÁ

- Instruções de controle

– A revolução da programação estruturada da década de 70 foi, em parte, uma reação à má legibilidade causada pelas limitadas instruções de controle das linguagens das décadas de 50 e 60.

– Uso indiscriminado de goto

LEGIBILIDADE

UNIVERSIDADE FEDERAL DO CEARÁ

- Tipos de dados

– Mecanismos adequados para definir tipos de dados

– Exemplo em C:

  - `int ligado = 1; // o significado dessa sentença não é claro`

– Exemplo em Java:

  - `boolean ligado = true; // o significado dessa sentença é claro`

LEGIBILIDADE

UNIVERSIDADE FEDERAL DO CEARÁ

- Projeto da sintaxe

– Formato dos identificadores:

  – Restringir os identificadores a tamanhos muito curtos piora a legibilidade.

  - Ex: Fortran77 com 6 letras e BASIC ANSI com uma letra ou letra e número

– Palavras especiais:

  - A legibilidade é fortemente influenciada pela forma das palavras especiais;

  - Ex: while, class e for

  - Como terminar sentenças? **end** ou **}**

    - Linguagens como C e descendentes

LEGIBILIDADE

– Forma e significado:

– A aparência da sentenças deve indicar parcialmente seu propósito

- Esse princípio é violado por duas construções de uma mesma linguagem idênticas ou similares na aparência, mas com significados diferentes, dependendo talvez do contexto;

- Ex: Em C a palavra reservada **static** depende do contexto

- Definição de uma variável dentro de uma função, significa que a variável é criada em tempo de compilação.

- Se for usada na definição de uma variável fora de todas as funções, significa que a variável é visível apenas no arquivo no qual sua definição aparece.

LEGIBILIDADE

UNIVERSIDADE FEDERAL DO CEARÁ

- É a medida da facilidade em que uma linguagem pode ser usada para criar programas para um domínio de problema escolhido.
- A maioria das características da linguagem que afetam a legibilidade também afetam a FE.
- Deve ser considerada no contexto do domínio de problema-alvo da linguagem

FACILIDADE DE ESCRITA

UNIVERSIDADE FEDERAL DO CEARÁ

- Simplicidade e ortogonalidade

  – Poucas construções, número pequeno de primitivas e um pequeno conjunto de regras para combiná-las

  - Programador pode projetar uma solução para um problema complexo após aprender apenas um conjunto simples de construções primitivas.

- Suporte à abstração

  - A habilidade de definir e usar estruturas ou operações complicadas de forma a permitir que muitos dos detalhes sejam ignorados

  - Abstração de processos: Uso de subprogramas;

  - Ex: Ordenar números;

  - Abstração de dados: Representar diversos dados por meio de uma referência ou estrutura;

  - Ex: Dados inteiros em um nó, dados de uma classe;

FACILIDADE DE ESCRITA

UNIVERSIDADE FEDERAL DO CEARÁ

- Expressividade
  - – Um conjunto de formas relativamente convenientes de especificar as operações
  - - Computações de uma forma conveniente, em vez de deselegante
  - – Número de operadores e funções pré-definidas
    - Exemplo: `cont++` em vez de `cont = cont + 1`, em C
    - Sentença for em vez de while em Java

CONFIABILIDADE

"Apresenta comportamento de acordo com suas especificações sob todas as condições."

- Verificação de tipos
  - Testes para detectar erros de tipos, por parte do compilador ou durante a execução
  - Fator importante na confiabilidade
  - Em tempo de execução é cara
    - Ex: Java em tempo de compilação
- Tratamento de exceções
  - Interceptar erros em tempo de execução, tomar medidas corretivas e continuar a execução
    - Exemplo de forte tratamento: Ada, C++ e Java
    - Exemplo de fraco tratamento: C e Fortran

CONFIABILIDADE

UNIVERSIDADE FEDERAL DO CEARÁ

- Utilização de apelidos (*Aliasing*)
  - Nomes distintos que podem ser usados para acessar a mesma célula de memória
    - Atualmente, é amplamente aceito que o uso de apelidos é um recurso perigoso em uma linguagem de programação
    - Trocar o valor apontado por um dos dois ponteiros modifica o valor referenciado pelo outro
- Legibilidade e facilidade de escrita
  - Uma linguagem que não oferece maneiras naturais para expressar os algoritmos requeridos irá necessariamente usar abordagens não naturais, reduzindo a confiabilidade.
  - Programas de difícil leitura complicam também sua escrita e sua modificação.

CUSTO

UNIVERSIDADE FEDERAL DO CEARÁ

- Treinar programadores para usar a linguagem
- Escrever programas (proximidade com o propósito da aplicação em particular)
- Compilar programas – necessita de otimização?
- Executar programas
- Sistema de implementação da linguagem: disponibilidade de compiladores gratuitos
- Confiabilidade baixa leva a custos altos – ex.: usina nuclear, medicina, aviação
- Manutenção de programas – correções, modificações para adição de funcionalidades

INFLUENCIA NA ESCOLHA DE UMA LP

UNIVERSIDADE FEDERAL DO CEARÁ

- Implementação
  - Disponibilidade quanto à plataforma
  - Eficiência: velocidade de execução do programa objeto
- Competência na LP
  - Experiência do programador
  - Competência do grupo envolvido
- Portabilidade
  - Necessidade de executar em várias máquinas

INFLUENCIA NA ESCOLHA DE UMA LP

UNIVERSIDADE FEDERAL DO CEARÁ

- Ambiente de programação
  - Ferramentas para desenvolvimento de software diminuem o esforço de programação
  - Bibliotecas

- Modelo de computação
  - Aplicação X modelo de computação
  - Por exemplo, para realização de busca heurística é adequado o Paradigma Lógico, para simulações, o Paradigma Orientado a Objeto

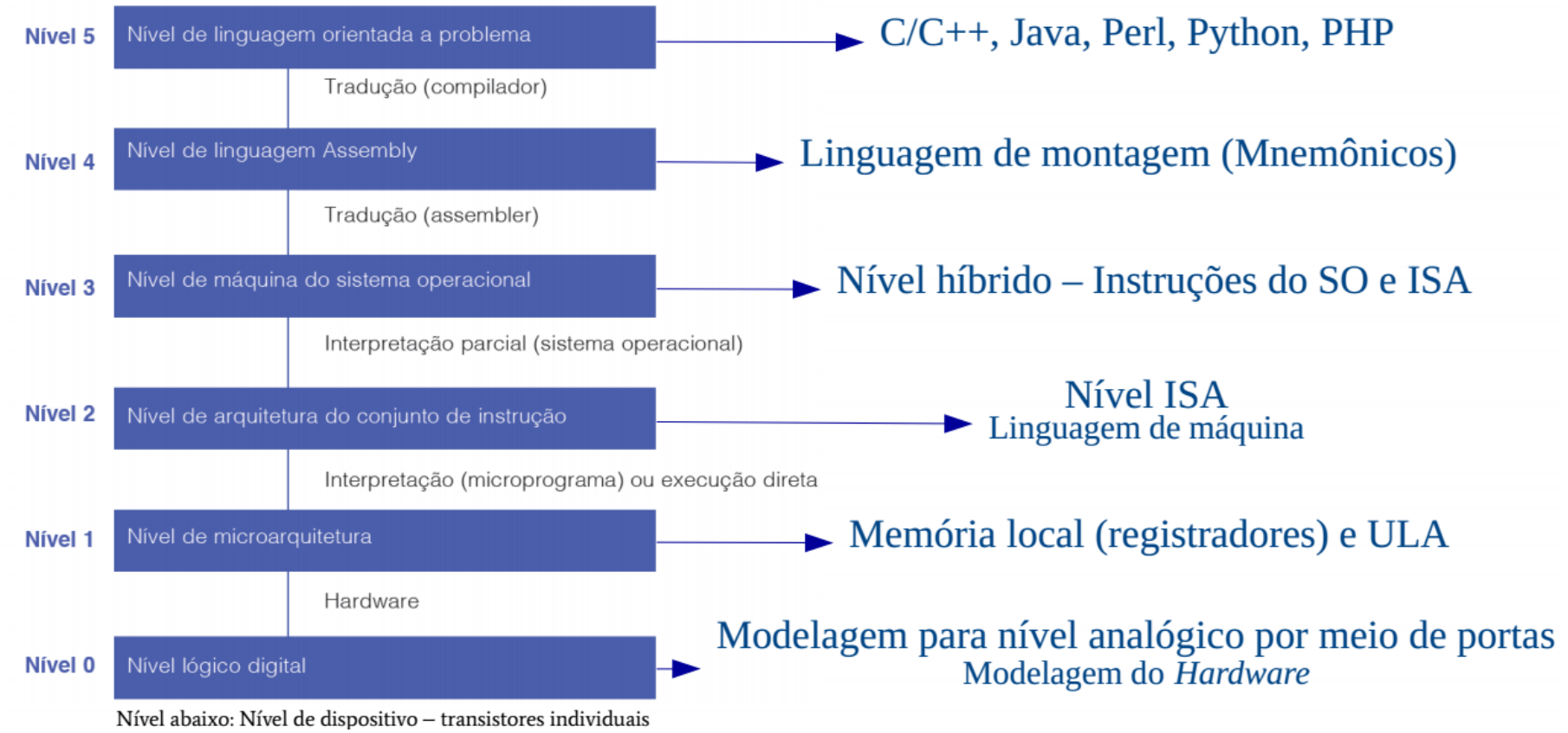
MÉTODOS DE IMPLEMENTAÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- Máquinas multiníveis;
- *Hardware* e *software* são logicamente equivalentes;
- "*Hardware* é apenas um *software* petrificado [Karen Panetta]

MÁQUINAS MULTINÍVEIS

UNIVERSIDADE FEDERAL DO CEARÁ



Nível abaixo: Nível de dispositivo – transistores individuais

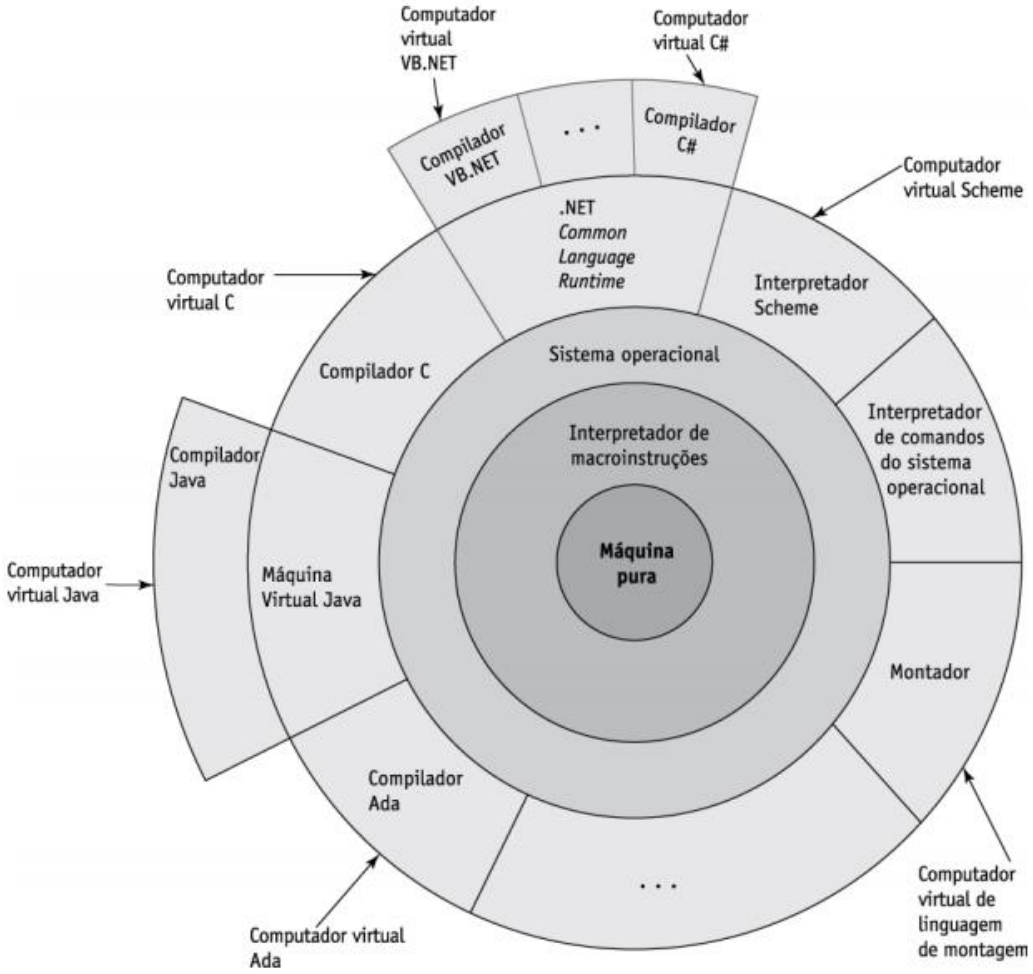
MÉTODOS DE IMPLEMENTAÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- Compilação
  - Programas são traduzidos para linguagem de máquina
- Interpretação pura
  - Programas são interpretados por outro programa chamado interpretador
- Sistemas de implementação híbridos
  - Um meio termo entre os compiladores e os interpretadores puros

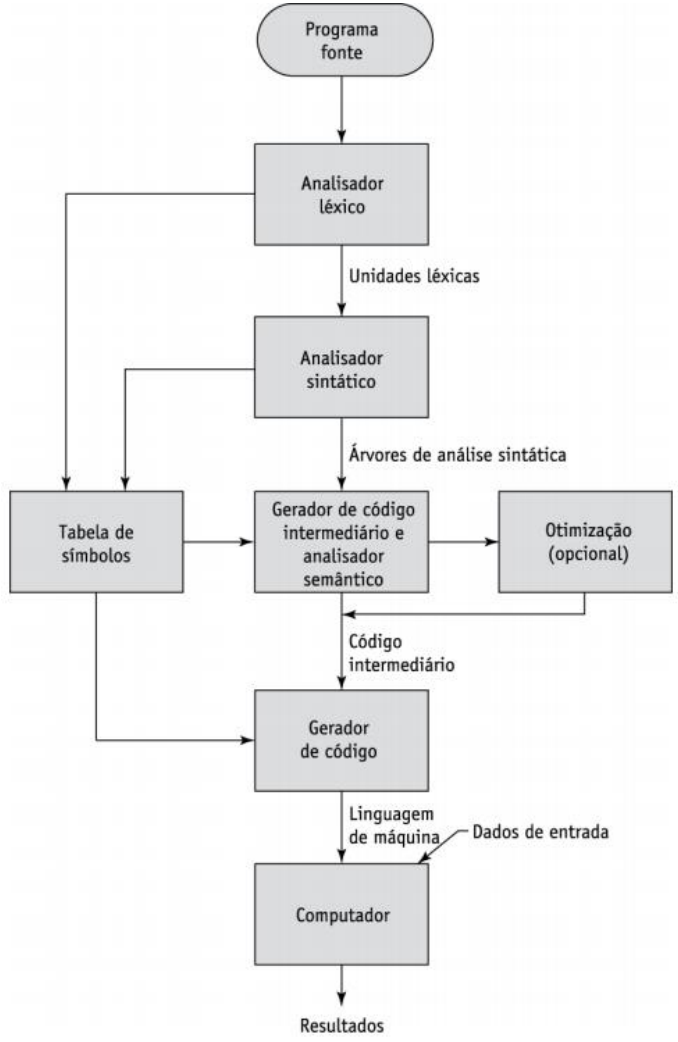
VISÃO EM CAMADAS DE UM COMPUTADOR

UNIVERSIDADE FEDERAL DO CEARÁ



COMPILAÇÃO

- Traduz programas (linguagem de fonte) em Código de máquina (linguagem de máquina)
- Tradução lenta, execução rápida
- Processo de compilação tem várias fases:
  - análise léxica: agrupa os caracteres do programa fonte em unidades léxicas
  - análise sintática: transforma unidades léxicas em árvores de análise sintática (*parse trees*), que representam a estrutura sintática do programa
  - análise semântica: gera código intermediário
  - geração de código: código de máquina é gerado



TERMINOLOGIA DE COMPILAÇÃO ADICIONAIS

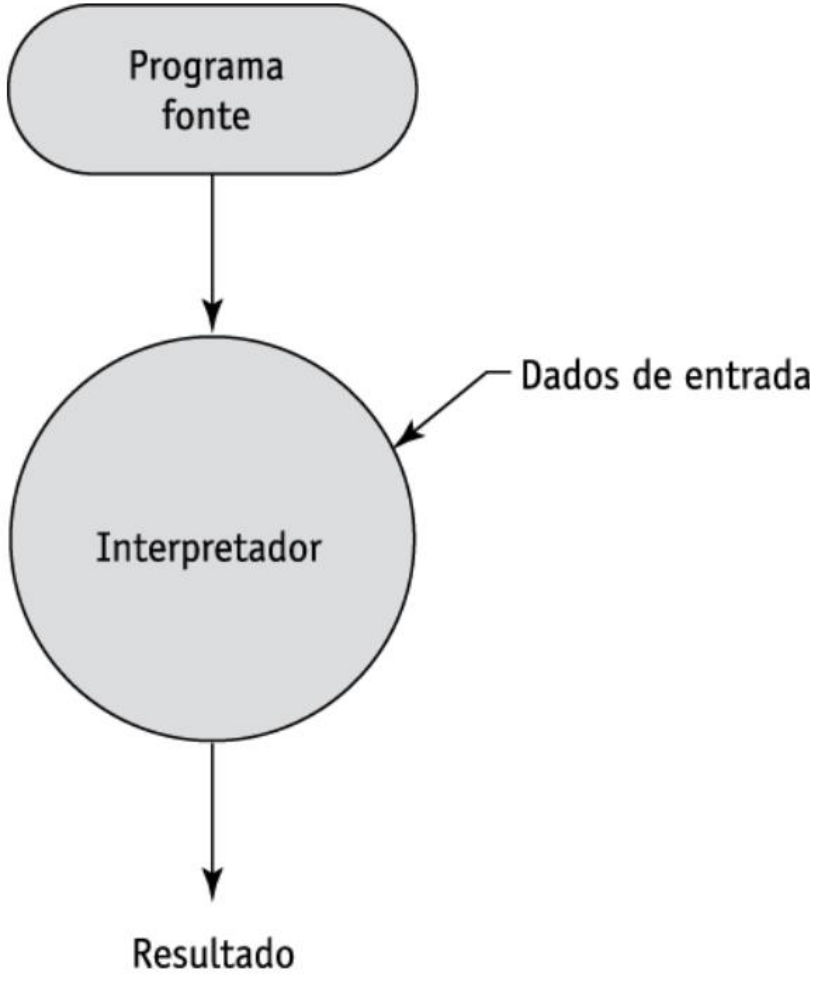
UNIVERSIDADE FEDERAL DO CEARÁ

- Módulo de carga (imagem executável): o código de usuário e de sistema juntos
- Ligação e carga: o processo de coletar programas de sistema, biblioteca ou de usuário e ligá-los aos programas de usuário
  - Realizada por um programa de sistema chamado de ligador (linker)

INTERPRETAÇÃO PURA

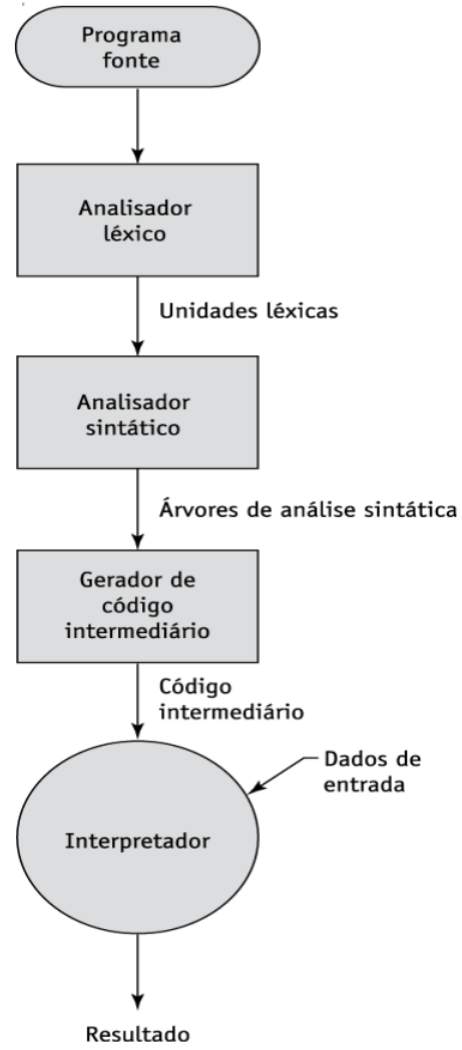
- Sem tradução
- Fácil implementação de programas
  (mensagens de erro em tempo de execução
  podem referenciar unidades de código fonte)
- Execução mais lenta (tempo de execução de
  10 a 100 vezes mais lento que sistemas compilados)
  - Decodificação das sentenças em linguagem de máquina,

  muito mais complexas do que as instruções de linguagem de máquina
- Geralmente requer mais espaço
  - Tabela de símbolos e programa fonte para fácil acesso e modificação
- Era raramente usada em linguagens de alto nível
- Volta significativa com algumas linguagens de
  scripting para a *Web* (como JavaScript e PHP)



SISTEMAS DE IMPLEMENTAÇÃO HÍBRIDOS

- Um meio termo entre os compiladores e os interpretadores puros;
- Uma linguagem de alto nível é traduzida para uma linguagem intermediária que permite fácil interpretação;
- Mais rápido do que interpretação pura;
  - Sentenças da linguagem fonte são decodificadas apenas uma vez



SISTEMAS DE IMPLEMENTAÇÃO HÍBRIDOS

UNIVERSIDADE FEDERAL DO CEARÁ

- Exemplos:

– Programas em Perl eram parcialmente compilados para detectar erros antes da Interpretação

– As primeiras implementações de Java eram todas híbridas; seu formato intermediário, bytecode, fornece portabilidade para qualquer máquina que tenha um interpretador de bytecodes e um sistema de tempo de execução associado (juntos, são chamados de Máquina Virtual Java – JVM)

- Atualmente as JVMs atuais executam bytecodes utilizando uma combinação de interpretação e compilação Just-In-Time(JIT)

- JIT: A JVM analisa os bytecodes à medida que eles são interpretados, procurando por *hot spots*

PRÉ-PROCESSADORES

UNIVERSIDADE FEDERAL DO CEARÁ

- As instruções de pré-processador são comumente usadas para especificar que o código de outro arquivo deve ser incluído
- É um programa que processa outro programa imediatamente antes de ele ser compilado.
- É essencialmente um programa que expande macros
- Um exemplo conhecido: pré-processador de C
  - expande `#include, #define` e macros similares

ATIVIDADE

- Pesquise sobre as LP citadas abaixo. Mostre seu breve histórico, características principais, importância, estrutura ou funções especificas, classificação quanto a seu paradigma.
  Insira um exemplo de código-fonte no relatório.
  - Algol, Pascal, Fortran
  - Basic, Cobol, Prolog
  - PL/I, Mumps, Clipper
  - Java, Assembly
  - Pesquise sobre linguagens desenvolvidas no Brasil
  - Fazer um relatório (word ou qualquer outro) e gerar PDF

REFERÊNCIAS

INTRODUÇÃO AOS PARADIGMAS DE PROGRAMAÇÃO. Prof. Joseph Soares Alcântara. UFC

Linguagens de Programação. Prof. Celso Olivete Júnior. UNESP