Paradigmas de Linguagens de Programação

Aspectos Preliminares

Cristiano Lehrer, M.Sc.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Motivação (1/6)

- Aumento da capacidade de expressar ideias:
  - Difícil conceituar estruturas quando não se pode descrevê-las.
  - Programadores são limitados pelas linguagens.
  - Aprender características de novas linguagens.
  - Simular, quando necessário, características de outras linguagens.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Motivação (2/6)

- Maior conhecimento para a escolha de linguagens apropriadas:
  - Programadores tendem a utilizar linguagens familiares, mesmo se está não é a mais apropriada para um novo projeto.
  - Familiarizar-se com as características de outras linguagens disponíveis, coloca o programador ou analista em melhor posição para fazer escolhas.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Motivação (3/6)

- Maior capacidade para aprender novas linguagens:
  - Programação é uma disciplina nova.
  - Está em contínua evolução.
  - Aprendizado contínuo é essencial, mas aprender nova linguagem pode ser um processo lento e difícil.
  - Entender os conceitos fundamentais de linguagem de programação ajudará ao futuro aprendizado.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Motivação (4/6)

- Entender melhor a importância da implementação:
  - Estudar os conceitos:
    - Estudar questões de implementação.
  - Porque a linguagem foi implementada de certa forma.
  - Usar uma linguagem de forma mais inteligente.
  - Conhecimento para correção de *bugs.*
  - Quando se conhece os mecanismos alternativos é possível escolher o mais eficiente.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Motivação (5/6)

- Aumento da capacidade para projetar novas linguagens:
  - Pode parecer impossível e remoto.
  - Ajudar desenvolvedores de linguagens:
    - Examinando e avaliando.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Motivação (6/6)

- Avanço global da computação:
  - Visão geral de computação justifica o estudo de Linguagens de Programação.
  - Avanço da computação é igual a avanço das linguagens.
  - As linguagens mais populares são as melhores:
    - Por que não JAVA ao invés do C++?

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Domínios de Programação (1/7)

- Aplicações científicas:
  - O computador foi inventado para aplicações científicas.
  - Estruturas de dados simples, mas grande quantidade de números em ponto flutuante.
  - ASSEMBLY:
    - Maior eficiência.
  - FORTRAN:
    - Primeira linguagem de alto nível para aplicações científicas.

Copyright © 2009/2024 • ybadoo.com.br

Domínios de Programação (2/7)

- Aplicações comerciais:
  - COBOL (1960).
  - Linguagens para este tipo de aplicação.
  - Entrada e saída mais elaboradas.
  - Microcomputador:
    - Planilhas eletrônicas e banco de dados.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Domínios de Programação (3/7)

- Inteligência artificial:
  - Aplicações de fronteira.
  - Uso de algoritmos não-determinísticos e computação simbólica e não apenas computação numérica.
  - LISP (1959) – programação funcional:
    - Primeira linguagem usada em aplicações de Inteligência Artificial.
  - PROLOG (70's) – programação lógica:
    - Conhecimento declarativo.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Domínios de Programação (4/7)

- Programação de sistemas:
  - Software de sistemas:
    - Sistemas operacionais e ferramentas de suporte à programação.
  - Devem ser eficientes.
  - IBM:
    - PL/S – um dialeto de PL/I.
  - UNIX:
    - Escrito em C.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Domínios de Programação (5/7)

- Sistemas Web:
  - Apresentação de conteúdo dinâmico:
    - Código junto com a tecnologia de apresentação.
    - Inicialmente:
      - PHP, Javascript.
  - Segurança.
  - Escalabilidade.
  - Desempenho.
  - Integração com sistemas existentes.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Domínios de Programação (6/7)

- Linguagem de *scripting*:
  - *Shell*:
    - KSH e CSH.
  - AWK:
    - Linguagem de geração de relatórios.
  - TCL:
    - Linguagem de *script* extensível.
  - PERL:
    - Combinação de *shell* e AWK.
  - Linguagem de 4ª geração:
    - Operação em banco de dados.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Domínios de Programação (7/7)

- Linguagens para propósitos especiais:
  - RPG:
    - Relatórios comerciais.
  - APT:
    - Ensino de ferramentas de máquinas programáveis.
  - GPSS:
    - Simulação de sistemas.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Critérios de Avaliação (1/2)

- Para avaliar características de uma linguagem é necessário ter critérios de avaliação.
- Principais critérios:
  - Legibilidade (*readability).*
  - Redigibilidade (*writability).*
  - Confiabilidade (*reliability).*

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Critérios de Avaliação (2/2)

Característica	Legibilidade	Redigibilidade	Confiabilidade
Simplicidade	x	x	x
Ortogonalidade	x	x	x
Tipos de dados	x	x	x
Projeto de sintaxe	x	x	x
Suporte para abstração		x	x
Expressividade		x	x
Verificação de tipos			x
Tratamento de exceções			x
Apelidos restritos			x

Copyright © 2009/2024 • ybadoo.com.br

Legibilidade (1/11)

- Facilidade com que programas podem ser lidos e entendidos.
- Simplicidade global:
  - Linguagem com grande número de componentes básicos é de difícil entendimento:
    - Programadores aprendem só um subconjunto.
  - Múltiplas formas de alcançar a mesma operação (complicador):
    - i = i + 1;
    - i += 1;
    - i++;
    - ++i;

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Legibilidade (2/11)

- Simplicidade global (continuação):
  - Sobrecarga do operador:
    - Característica útil, mas exige racionalidade no uso.
    - Símbolo de +:
      - inteiros, reais, vetores, concatenação de strings.
  - Muita simplicidade = pouca legibilidade (ASSEMBLY):
    - Faltam estruturas de controle e de dados.
    - Instruções são muito simples.
    - Requer várias instruções para atingir uma operação de alto nível.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Legibilidade (3/11)

- Ortogonalidade:
  - Um conjunto relativamente pequeno de primitivas podem ser combinadas (num número relativamente pequeno de modos) para formar estruturas de dados e de controle.
  - ASSEMBLY do Mainframe IBM:
    - A reg1, mem {reg1 = reg1 + memória}
    - AR reg1, reg2 {reg1 = reg1 + reg2
  - ASSEMBLY do Supermini VAX:
    - ADDL oper1, oper2 {oper2 = oper1 + oper2}

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Legibilidade (4/11)

- Ortogonalidade (continuação):
  - Quanto mais ortogonal, menor o número de regras de exceção:
    - Poucas regras:
      - Mais fácil de aprender, ler e entender.
  - Linguagem funcional:
    - Boa combinação de simplicidade e ortogonalidade.
    - Tudo são funções aplicadas às listas.
    - LISP:
      - (ADD X 5 (SUB Y Z (DIV 5 7) 4) W)

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Legibilidade (5/11)

- Ortogonalidade (continuação):
  - Falta de ortogonalidade → C
    - Possui dois tipos de dados estruturados, vetores e registros (struct), mas somente registros podem ser retornados de funções.
    - Parâmetros são passados por valor, a menos que seja um vetor, que é passado por referência.
  - Ortogonalidade até demais → ALGOL 68
    - Objetos de qualquer tipo são permitidos como parâmetros e resultados de subprogramas.
    - Podem surgir programas extremamente difíceis de ler.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Legibilidade (6/11)

- Exemplo de falta de ortogonalidade em C:

```c
typedef struct par {
  int primeiro;
  int segundo;
} par;

void f(par x) {
  x.primeiro = 10;
}

void test_f() {
  par x = {1, 2};
  // passagem de parâmetro
  // por valor
  f(x);
  assert(x.primeiro == 1);
}
```

```c
void g(int x[2]) {
  x[0] = 10;
}

void test_g() {
  int x[2] = {1, 2};
  // vetores não podem ser
  // passados como parâmetro
  // por valor!
  g(x);
  assert(x[0] == 10);
}
```


Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Legibilidade (7/11)

- Ortogonalidade (continuação):
  - “Se tentarmos conseguir simplicidade através da generalização da linguagem, poderemos obter programas que, por sua própria concisão e falta de redundância, escaparão de nossa capacidade intelectual limitada .... A chave, então, se encontra não tanto na minimização do número de características básicas da linguagem, mas sim em manter os recursos incluídos fáceis de entender em toda consequência de seus usos e livres de interações inesperadas quando combinadas” (Niklaus Wirth, 1975).

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Legibilidade (8/11)

- Instruções de controle:
  - goto:
    - Reduz a legibilidade de programas.
  - FORTRAN e BASIC:
    - Faltam estruturas de controle.
  - Linguagens mais recentes:
    - Eliminaram a necessidade de goto.
    - Pode-se ler um programa de cima a baixo sem saltos.
  - PASCAL tinha goto, o MODULA-2 não.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Legibilidade (9/11)

- Tipos de dados e estruturas:
  - Facilidades de definição de tipos e estruturas de dados ajudam na legibilidade:
    - Necessidade de usar um tipo inteiro ao invés de um tipo lógico:
      - soma_e_muito_grande = 1;
      - soma_e_muito_grande = true;
    - Uso de registros pode facilitar o entendimento:
      - Formulários de empregados.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Legibilidade (10/11)

- Considerações sobre a sintaxe:
  - Formas identificadoras:
    - Tamanho pequeno dificulta o entendimento:
      - FORTRAN → máximo de seis caracteres.
      - ANSI BASIC → somente uma letra ou uma letra e um dígito.
  - Palavras especiais:
    - Grupos de instruções/instruções compostas:
      - begin ... end
      - Chaves – { }
      - end if; end loop;
  - Palavras especiais podem ser usadas como identificadores:
    - FORTRAN 77 → do e end são nomes de variáveis legais.

Copyright © 2009/2024 • ybadoo.com.br

Legibilidade (11/11)

- Forma e significado (sintaxe e semântica):
  - Semântica deve seguir diretamente a sintaxe.
  - static do C é dependente do contexto:
    - Se for usada dentro de uma função, indica que a variável é criada em tempo de compilação.
    - Se for criada fora de todas as funções, indica que esta variável é visível somente neste arquivo.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Redigibilidade (1/5)

- É a medida de quão facilmente uma linguagem pode ser usada para criar programas para o domínio do problema escolhido.
- O que afeta a legibilidade também afeta a redigibilidade:
  - Ao escrever, estamos sempre relendo.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Redigibilidade (2/5)

- Deve ser considerada no contexto do domínio do problema alvo da linguagem:
  - COBOL
    - Relatórios financeiros com formatos complexos.
  - FORTRAN
    - Cálculos utilizando números complexos.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Redigibilidade (3/5)

- Simplicidade e ortogonalidade:
  - Grande número de construções:
    - Falta de uso de algumas que poderiam ser mais elegantes e/ou eficientes.
  - Um pequeno número de primitivas e um conjunto de regras consistentes para combiná-las é muito melhor que ter um grande número de primitivas.
  - Muita ortogonalidade pode comprometer a redigibilidade:
    - Erros podem não ser detectados porque várias combinações podem ser legais.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Redigibilidade (4/5)

- Suporte para abstração:
  - Abstração:
    - Habilidade de se definir e usar estruturas e/ou operações complicadas de tal forma que detalhes sejam ignorados.
  - É um conceito chave no projeto das linguagens modernas.
  - As linguagens podem suportar duas formas de abstração:
    - Processos:
      - Utilização de subprogramas (funções ou procedimentos).
    - Dados:
      - Estruturas de dados.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Redigibilidade (5/5)

- Expressividade:
  - Refere-se a várias características diferentes.
  - APL
    - Programas pequenos com a utilização de operadores poderosos.
  - C:
    - Mais conveniente usar i++ do que i = i + 1;
  - Operadores lógicos proporcionam boa forma de expressar condições.
  - for torna a escrita de contadores de *loops* mais fácil do que usar while.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Confiabilidade (1/4)

- Um programa é dito ser confiável se ele executa suas especificações sobre qualquer condição.
- Verificação de tipos:
  - Teste de erros de tipos, sejam em compilação ou em execução.
  - Examinar tipos durante a compilação é muito melhor.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Confiabilidade (2/4)

- Manipulação de exceções:
  - É a habilidade de um programa de interceptar erros de tempo de execução, tomar medidas corretivas e continuar o processamento.
- Apelidos (*aliasing*):
  - É ter dois nomes para a mesma área de memória:
    - Ponteiros.
    - Referências.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Confiabilidade (3/4)

- Uso de apelidos em C++, Java e Rust

C++

```
vector<int> v = {10, 20, 30};
int soma = 0;
for (int &x : v) {
  soma += x;
  if (x == 20)
    v.push_back(1);
}
assert(soma == 61);
```

Pode ou não falhar… x pode referenciar memória desalocada

Java

```
ArrayList<Integer> lista =
new ArrayList<>(
asList(10, 20, 30));
int soma = 0;
for (Integer x : lista) {
  soma += x;
  if (x == 20)
    lista.add(1);
}
assert soma == 61;
```

Falha na execução: lista.add gera o erro java.util.ConcurrentModificationException

Rust

```
let mut v = vec![10, 20, 30];
let mut soma = 0;
for &x in &v {
 soma += x;
 if x == 20
  v.push(1);
}
assert_eq!(soma, 61);
```

Falha na compilação: cannot borrow v as mutable because it is also borrowed as immutable

Copyright © 2009/2024 • ybadoo.com.br

Confiabilidade (4/4)

- Legibilidade e redigibilidade:
  - Facilidade de escrever:
    - Maior chance de estar correto.
  - Métodos não naturais são mais comuns de estarem com erros.
  - Legibilidade afeta a confiabilidade tanto na escrita quanto na manutenção.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Custos

- Custo de treinar os programadores.
- Custo de escrever programas em certa linguagem:
  - Bons ambientes facilitam a programação.
- Custo de compilar um programa.
- Custo de execução do programa.
- Custo dos compiladores.
- Custo de manutenção dos programas.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Influências sobre o Projeto da Linguagem (1/4)

- Adicionalmente aos fatores levantados nos critérios de avaliação, outros fatores influenciam no projeto de linguagens.
  - Arquitetura do computador:
    - A maioria das linguagens foram projetadas em torno na arquitetura de von Neumann.
    - Tais linguagens são chamadas imperativas.
    - As principais características dessas linguagens são:
      - Variáveis (células de memória).
      - Instruções de atribuição.
      - Forma iterativa de repetição.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Influências sobre o Projeto da Linguagem (2/4)

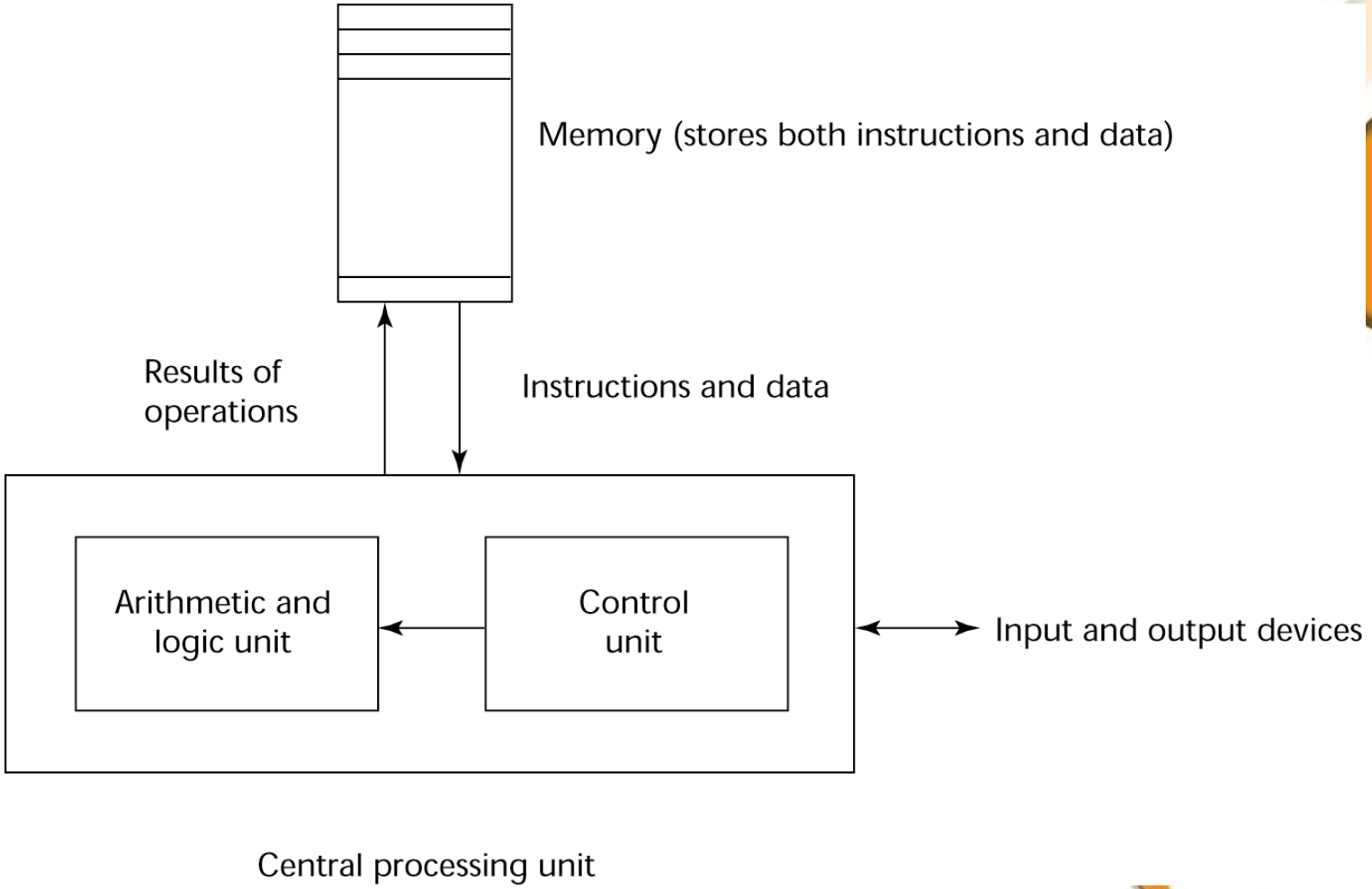
- Arquitetura do computador (continuação):
  - Iterações são muito eficientes porque as instruções são armazenadas em células adjacentes de memória.
  - Embora recursão possa ser mais natural, a arquitetura desencoraja sua utilização por razões de eficiência.
  - Linguagens funcionais não tomarão o lugar das linguagens imperativas, a menos que surja uma outra arquitetura.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Influências sobre o Projeto da Linguagem (3/4)

- Arquitetura do computador (continuação):



Copyright © 2009/2024 • ybadoo.com.br

Influências sobre o Projeto da Linguagem (4/4)

- Metodologias de Programação:
  - 1950 a 1960:
    - Aplicações simples e aborrecimento sobre a eficiência das máquinas.
  - 1960 a 1970:
    - Eficiência tornar-se importante, legibilidade, estruturas de controle melhor.
  - 1970 a 1980:
    - Abstração de dados.
  - 1980 em diante:
    - Programação orientada a objetos.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Categorias de Linguagens

- É comum a seguinte classificação hierárquica:
  - Imperativas
    - Procedurais (Fortran, Pascal, C, …)
      - Aplicar algumas instruções a conjunto de dados.
    - Orientada a Objetos (Smalltalk, Eifel, Java, C++, …)
      - Relacionada com a procedural.
  - Declarativas
    - Funcionais (Lisp/Scheme, Haskell, ML, …)
      - Aplicar uma função (no sentido matemático) a alguns parâmetros.
    - Lógicas (Prolog)
      - Realizar uma sequência de passos dedutivos tendo-se um conjunto de propriedades entre dados e relacionamento entre dados.
- Linguagens de marcação (*markup*) como HTML não são linguagens de programação, pois não especificam computações, apenas a aparência geral de documentos.

Copyright © 2009/2024 • ybadoo.com.br

Trade-offs no Projeto da Linguagem

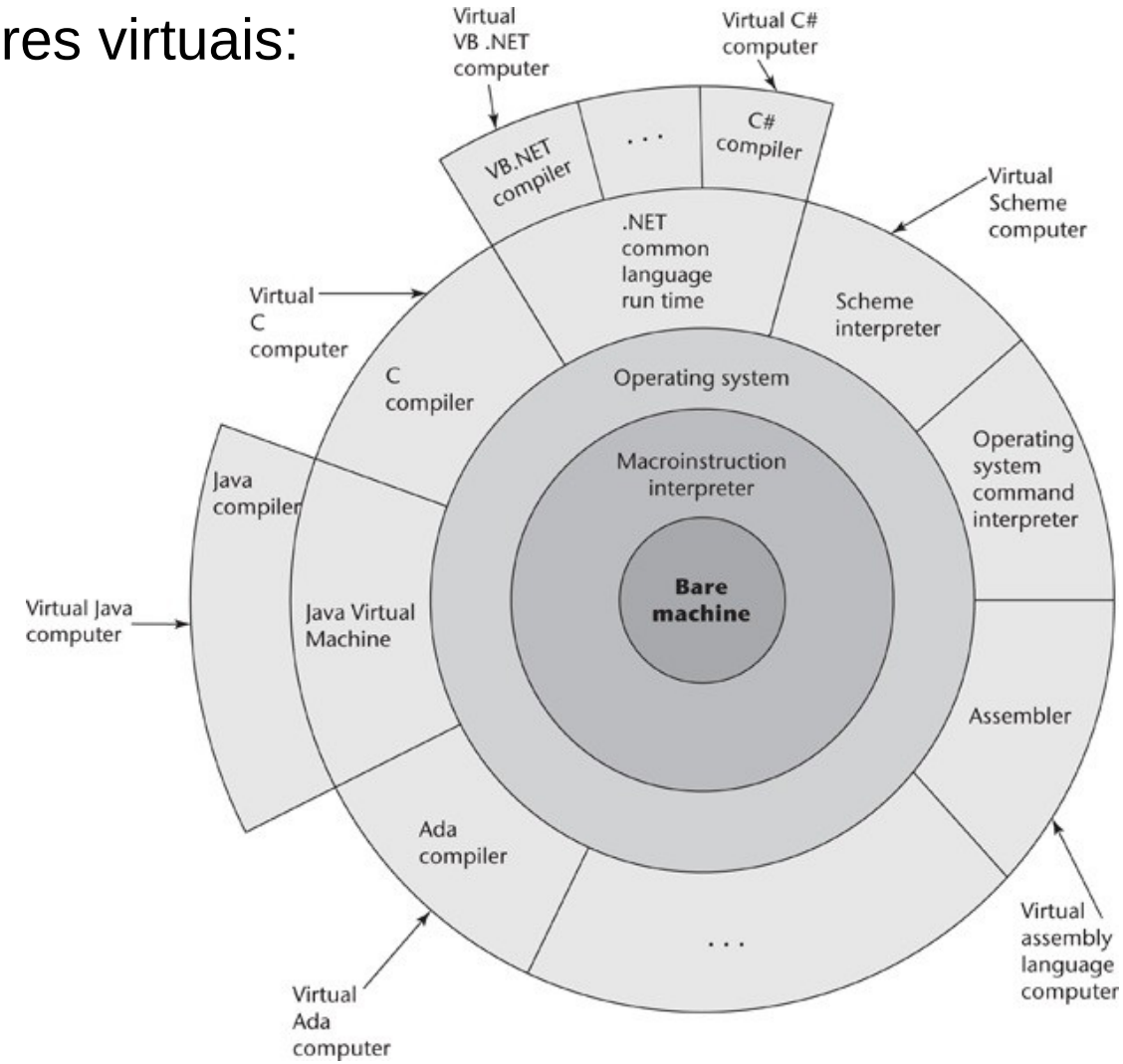
- Confiabilidade versus custo de execução.
- Capacidade de escrita versus legibilidade.
- Flexibilidade versus segurança.

Copyright © 2009/2024 • ybadoo.com.br

ybadoo

Métodos de Implementação (1/4)

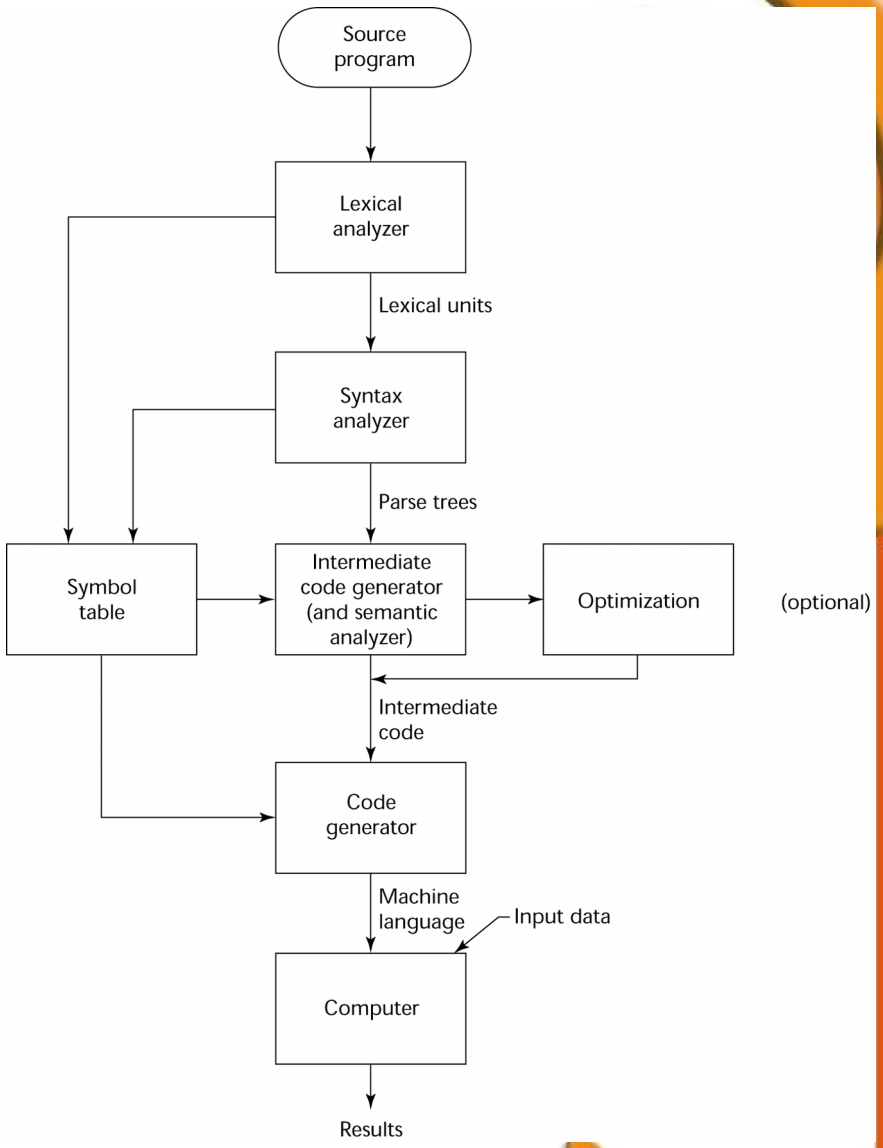
- Computadores virtuais:



Copyright © 2009/2024 • ybadoo.com.br

Métodos de Implementação (2/4)

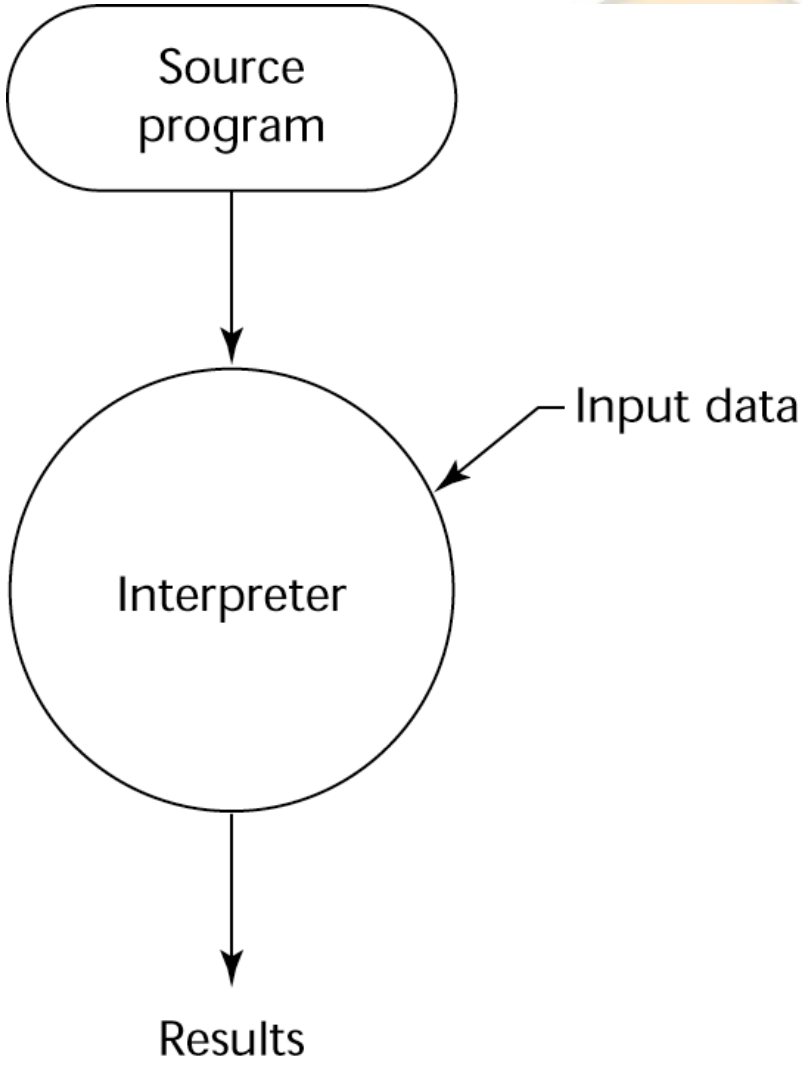
- Compilação:
  - Tradução de programa de alto nível para código de máquina.
  - Tradução é lenta.
  - Execução é rápida.



Copyright © 2009/2024 • ybadoo.com.br

Métodos de Implementação (3/4)

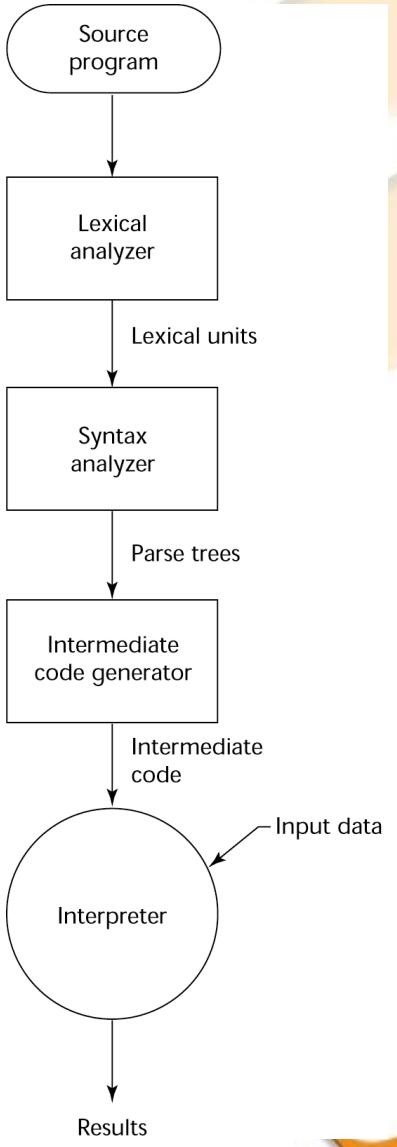
- Interpretação pura:
  - Sem tradução.
  - Execução lenta.
  - Raramente utilizado.



Copyright © 2009/2024 • ybadoo.com.br

Métodos de Implementação (4/4)

- Sistemas de implementação híbridos:
  - Custo de tradução pequeno.
  - Velocidade de execução média.



Copyright © 2009/2024 • ybadoo.com.br

Ambientes de Programação

- A coleção de ferramentas utilizadas no desenvolvimento de software:
  - UNIX
    - Sistema operacional e coleção de ferramentas antigas.
  - Borland C++
    - Ambiente de desenvolvimento para C e C++.
  - SMALLTALK
    - Linguagem e ambiente de desenvolvimento integrados.
  - Eclipse
    - Ambiente de desenvolvimento para múltiplas linguagens de programação:
      - Java
      - C++
      - PHP

Copyright © 2009/2024 • ybadoo.com.br

ybadoo