

CRITERIA	Characteristic		
	READABILITY	WRITABILITY	RELIABILITY
Simplicity/orthogonality	•	•	•
Control structures	•	•	•
Data types and structures	•	•	•
Syntax design	•	•	•
Support for abstraction	•	•	•
Expressivity	•	•	•
Type checking	•	•	•
Exception handling	•	•	•
Restricted aliasing	•	•	•

**Table 1.1** Language evaluation criteria and the characteristics that affect them.

## Criterios de avaliação de uma linguagem

- Legibilidade (Readability)
  - Qual o grau de conformidade com as especificações sob todas as condições?
  - Capacidade de escrita (Writability)
    - Qual facilmente uma linguagem pode ser usada para criar programas?
    - Qual facilmente um programa pode ser lido e entendido?
- Capacidade de leitura (Readability)
  - Qual facilmente uma linguagem pode ser usada para criar
- Confidabilidade (Reliability)
  - Qual o grau de conformidade com as especificações sob todas as condições?
- Custo
  - Qual o custo final de uma linguagem é função de suas características?

## Criterios de avaliação de uma linguagem

- Todas combinações possíveis são legíveis e significativas
- Conjunto relativamente pequeno de maneiras primitivas pode ser combinado em um número
- Mesmo operador para mais de um significado
- Sobrecarga (overloading) de um operador
- Ortogonalidade

## Legibilidade (Readability)

- Mais de uma maneira para realizar uma operação
- Multiplicidade de recursos
- Linguagens com poucos elementos básicos são mais particulares
- Por exemplo: Em C, pode-se incrementar uma variável
- Intera simples das seguintes maneiras:
  - `count++`
  - `count = count + 1`
  - `count += 1`

## Legibilidade (Readability)

um ponto final...  
■ Quando esse é o caso o programador se obriga a retornar

podem retornar arrays

- Funções em C podem retornar registros mas NÃO
  - uma linguagem de alto nível é:
- Um exemplo de ausência de ortogonalidade em

## Ausência de Ortogonalidade

instrução de adição do VAX  
ortogonal. Uma unica  
operando pode usar  
registros ou  
memória como  
operando como

Computador IBM (Grande porte)  
A Reg1, célula de memória  
Reg1 → conteúdo(Reg1) + conteúdo(célula de memória)  
Semântica:  
AR Reg1, Reg2  
A Reg1, célula de memória  
Reg1 → conteúdo(Reg1) + conteúdo(Reg2)  
Semântica:  
ADD operando1, operando2  
operação de adição do VAX

- Adição de inteiros (32 bits) residentes em memória/registradores (Substituir um dos valores pela soma)

## Exemplo de Ortogonalidade

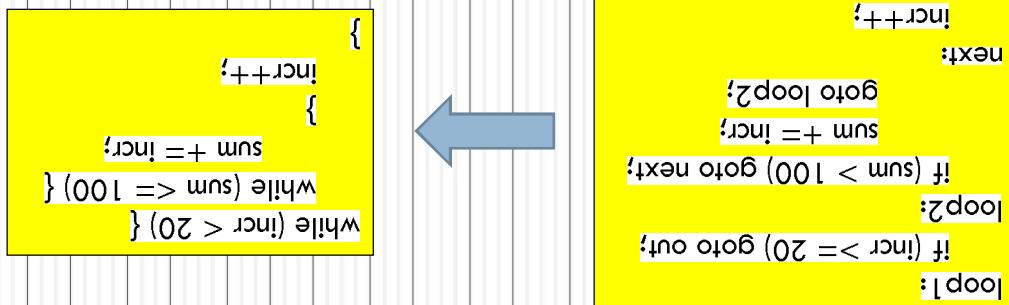
Note a diferença na clareza de interpretação simplesmente por ter ou não o tipo de dado booleano pre-definido...

timeOut = true



- Tipos de dados e estruturas
- Dispõe de recursos adequados para definição

## Legibilidade (Readability)



- Existência de estruturas de controle apropriadas
- Instruções de controle

## Legibilidade (Readability)

- Inclusão do `for` em muitas linguagens modernas
- `count++` é mais simples que `count = count + 1`
- Exemplos:
- Conjunto conveniente de maneiras de especificar operadores
- Expressividade
  - Ignorando máximo possível de detalhes
- Capacidade de definir/usar estruturas/operações complexas
- Abstração
  - Menor quantidade possível de cada
- Construtores, primitivas e conjunto de regras para combinações
- Simplicidade e ortogonalidade

## Capacidade de Escrita (Writability)

- Projeto instruções de forma que sua aparência indique sua finalidade
- Forma e significado
  - Podem ou não ser usadas como identificadores
- Unicidade
  - Exemplos: `while`, `class`, `for` e `begin-end`
- Representatividade
  - Restringir o tamanho dos identificadores pode prejudicar a legibilidade
- Palavras-chave
  - Considerações sobre a sintaxe
- Identificadores
  - Restringir o tamanho dos identificadores pode prejudicar a legibilidade
- Forma e significado
  - Podem ou não ser usadas como identificadores
- Unicidade
  - Exemplos: `while`, `class`, `for` e `begin-end`
- Representatividade
  - Restringir o tamanho dos identificadores pode prejudicar a legibilidade
- Palavras-chave
  - Considerações sobre a sintaxe

## Legibilidade (Readability)

- Tanto na escrita quanto na manutenção
- Legibilidade afeta confiabilidade a legibilidade
- Usarão, necessariamente, formas não-naturais, prejudicando algoritmos
- Linguagem sem suporte natural para expressar (Writability)
- Legibilidade (Readability) e Capacidade de Escrita

## Confiabilidade (Reliability)

- referenciando mesma célula de memória
- Presente de mais de um método, ou nome, distintos
- Apelidos (Aliasing)
- execução e porém praticamente corretivas
- Capacidade de interceptar erros em tempo de manipulação de Exceções
- Possibilidade de identificar erros de tipos
- Verificação de tipos

## Confiabilidade (Reliability)

línguagem

- A precisão é a completnude da definição oficial da
- Boa definição (Well-definedness)
- Seu uso em uma gama de aplicações
- Generalidade
- Uma implementação para outra
- Quão facilmente um programa pode ser movido de
- Portabilidade

## Outros critérios de avaliação

Custo

- Inversamente proporcional aos custos
- Confidencialidade
- Existência de compiladores gratuitos
- Sistema de implementação da línguagem
- Compilação e execução de programas
- Escrita e manutenção de programas
- Treinamento dos programadores
- Associado com tempo gasto

- Instruções e dados trafegam da memória para a CPU
- Memória separada da CPU
- Dados e programas armazenados na memória
- Totalmente relacionada ao paradigma imperativo

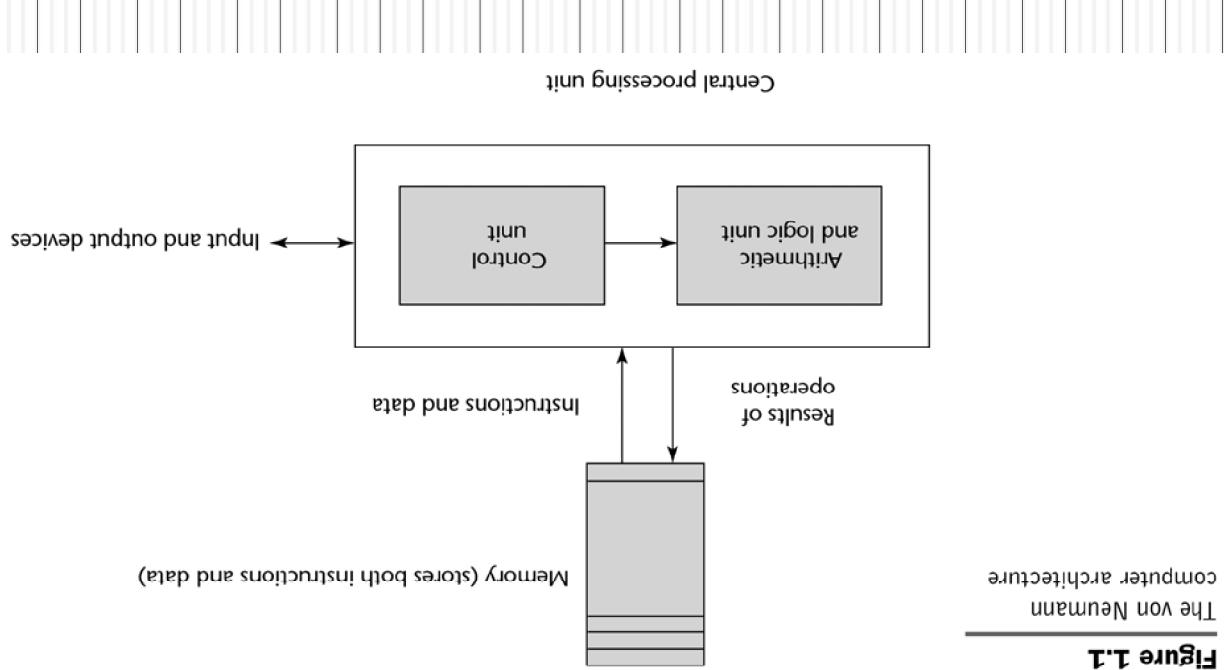
## Arquitetura de von Neumann

- Exemplo: desenvolvimento orientado a objetos
- Paradigmas e linguagens de programação
- Novas metodologias de desenvolvimento criam novos
- Metodologias de programação
- Maioria dos paradigmas assume a arquitetura de von Neumann
- Arquitetura do Computador

## Infleções sobre o projeto de uma linguagem

- ❑ Fator limitante básico da velocidade dos computadores
- ❑ Esta religião é denominada **garrafa de von Neumann**
- ❑ Istruções geralmente são executadas mais rapidamente do que podem ser transferidas
- ❑ Velocidade de um computador determinada pelo processador e seu memória
- ❑ Velocidade de conexão entre a memória do computador e seu processador determina a velocidade de um computador

## Garrafa de Von Neumann



## Arquitetura von Neumann

Figure 1.1

The von Neumann computer architecture

- Década de 50 e início da de 60
  - Preocupação era a eficiência da máquina
  - Aplicações simples
  - Foco agora são as pessoas: legibilidade, melhores estruturas de controle
  - Projeto top-down e refinamento passo-a-passo
  - Programação estruturada
  - Projeto top-down e refinamento passo-a-passo
  - Orientado para processo ⇔ orientada a dados
  - Final da década de 70
  - Abstração de dados + herança + polimorfismo
  - Medidas da década de 80
    - Abstração de dados
    - Programação orientada a objetos
    - Final da década de 90
    - Foco agora são as pessoas: legibilidade, melhores estruturas de
    - Foco agora são as pessoas: legibilidade, melhores estruturas de
    - Foco agora são as pessoas: legibilidade, melhores estruturas de
    - Foco agora são as pessoas: legibilidade, melhores estruturas de

## metodologias de programação influências sobre as

- O argumento de von Neumann é tido como o mais importante precursor desse novo paradigma
- Mais de uma CPU
- Memórias cache
- Segundo critério
- Consistência da memória
- Estagnação
- E, apesar de continuar com o conceito de memória
  - (imperativo), novos problemas surgem

## Paradigma Concorrente e Paralelo

## Custo/benefício no projeto

### Confiabilidade versus Custo de Execução

■ Exemplo: Java requer que todas as referências a

vetores sejam verificadas para garantir que os índices estejam dentro dos limites. No entanto, isso aumenta o

custo de execução do programa (já em C...).

## Custo/benefício no projeto

### Confiabilidade versus Custo de Execução

■ Exemplo: Java requer que todas as referências a

vetores sejam verificadas para garantir que os índices estejam dentro dos limites. No entanto, isso aumenta o

custo de execução do programa (já em C...).

## Custo/benefício no projeto

### Readability versus Writability

■ Exemplo: APL provê muitos operadores poderosos (e

uma grande quantidade de novos símbolos), permitindo que computações complexas sejam escritas em programas compactos, porém isso dificulta a leitura (já

em COBOL...).

## Aspectos de implementação

- Combinagão entre compiladores e interpretadores
- Implementação híbrida
- Executável temporário
- Programas são interpretados, gerando um código
- Interpretagão pura
- Programas tornam-se linguagem de máquina
- Compilagão

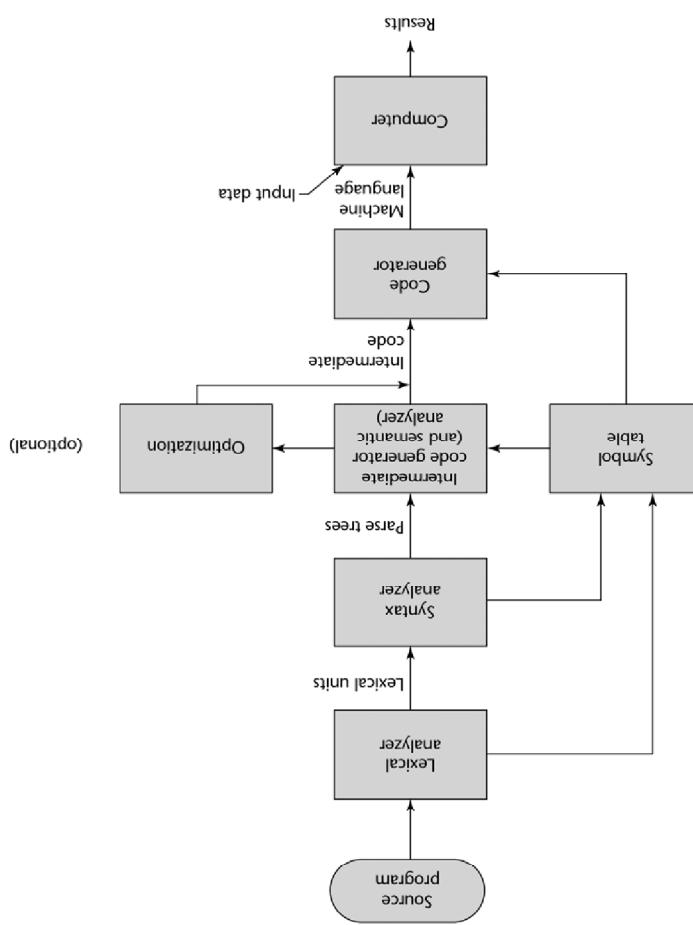
## Métodos de implementação

- Desafio atual de pesquisas acadêmicas: Verificar confiabilidade de programas que usam ponteiros
- Exemplo: Ponteiros em C++ são poderosos e muito flexíveis. Porém, a confiabilidade dos programas diminui
- Writability (flexibility) versus reliability

## Custo/benefício no projeto da linguagem: Critérios conflitantes

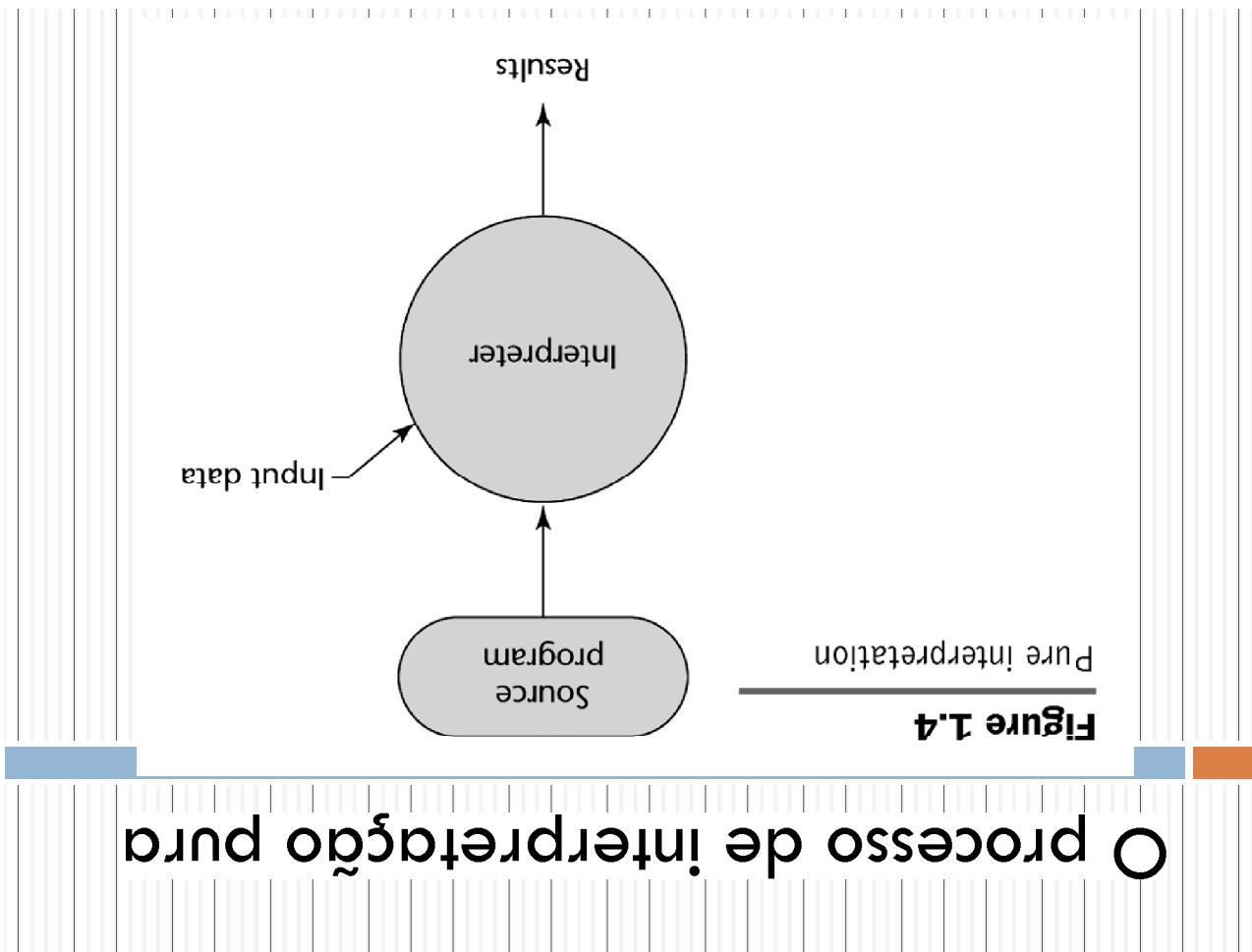
# O processo de compilação

The compilation process



- Linguagem-fonte é traduzida em linguagem de máquina
- Traduzido é lenta, mas executado é rápido
- Processo possui várias fases
- Análise léxica
- Caracteres de programa são identificados como unidades léxicas
- Unidades léxicas relacionados originalmente árvores sintáticas (representam estrutura do programa)
- Análise sintática
- Caracteres de programa são identificados como unidades léxicas
- Unidades léxicas relacionados árvores sintáticas
- Análise semântica
- Cada estrutura recebe um significado (Código intermediário)
- Geração de código
- Código intermediário é mapeado na máquina em questão

## Compilação



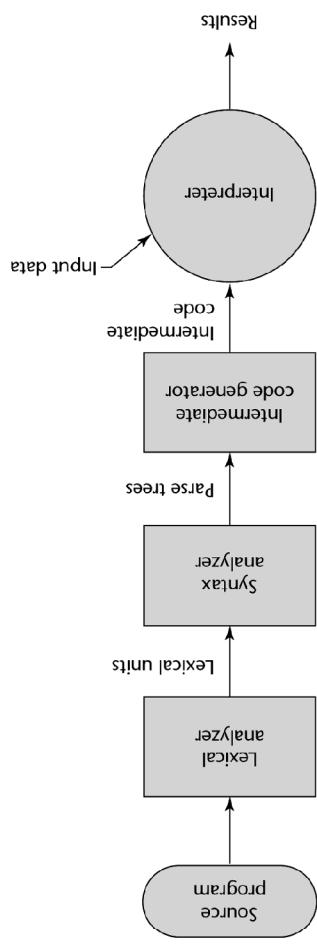
## O processo de interpretação pura

- Sem tradução
- Fácil implementação de programas
- Erros de execução mais fácil e rapidamente mostrados
- Execução lenta
- De 10 a 100 vezes menor que versões compiladas
- Geralmente requer mais espaço
- Cada vez mais raro em linguagens de alto-nível
- Embora não seja uma linguagem simples, JavaScript é puramente interpretado

## Interpretação Pura

# O processo de implementação híbrida

Figure 1.5  
Hybrid implementation system



- Combinação entre compilador e interpretador puro
- Programa em linguagem de alto-nível é traduzido para uma linguagem intermediária de fácil interpretação
- Mais rápido que interpretação pura e menos que compilada
- Exemplos
- Perl é parcialmente compilado para detectar erros antes da interpretação
- Implementações iniciais de Java eram híbridas
- Forma intermediária, bytecode, fornecia portabilidade a qualquer máquina que tivesse interpretador de bytecode e sistema de run-time (conjunto chamado de Java Virtual Machine)

## Híbridos

## Sistemas de implementação

- Ambientes gráficos e complexo usado para programar em C#, Visual BASIC.NET, JavaScript, J# ou C++
- Microsoft Visual Studio.NET
- Ambientes integrados de desenvolvimento para Java
- Borland JBuilder/Eclipse
- KDE ou GNOME, abstraindo comandos textuais
- Atualmente, seu uso se dá através de GUI (exemplos: CDE, Gnome, KDE)
- Sistema operacional e conjunto de ferramentas
- UNIX
- Coleção de ferramentas usadas para desenvolver software

## Ambientes de Programação

- Ferramentas que manipulam linguagem-fonte
- Instruções (macros) para manipulação de texto e arquivos
- Macros podem incluir arquivos em outros (#include) ou criar apelidos para expressões complexas (#define)
- Processam um programa antes de sua compilação

## Pre-processadores