Linguagens de Programação

Roberto Willrich
INE- CTC-UFSC
E-Mail: willrich@inf.ufsc.br
URL: http://www.inf.ufsc.br/~willrich

Linguagens de Programação

- Conteúdo
  - **Software e Hardware**
  - **Tipos de Softwares**
  - **Níveis de Linguagem de Programação**
  - **Etapas para Geração de um Programa**
  - **Paradigmas de programação**
  - **Linguagens Interpretadas e Compiladas**
  - **Exemplos de Linguagens de Programação**

Programação de Computadores

- Sistema Computacional
  - **Termo mais abrangente que computador**
    - Inclui qualquer máquina baseada em processador
      - Características diferentes tanto a nível de arquitetura e linguagem de programação
  - **Visto como uma associação de**
    - **Hardware**
      - está associado à parte física do sistema (os circuitos e dispositivos) que suporta o processamento da informação;
    - **Software**
      - corresponde ao conjunto de programas responsáveis pela pilotagem do sistema para a execução das tarefas consideradas

Programação de Computadores

- Classificação dos softwares quanto ao tipo de serviço por ele realizado
  - **software de sistema (ou sistema operacional)**
    - capaz de oferecer ao usuário, ou a outros softwares, facilidades de acesso aos recursos do computador
    - através de comandos ou serviços especiais a nível de um programa
    - administra os arquivos, controla periféricos e executa utilitários.
  - **software utilitário**
    - programas desenvolvidos por especialistas ou mesmo por usuários experimentados
    - tem por objetivo facilitar a realização de determinadas atividades correntes no uso dos computadores
      - detecção e eliminação de vírus, programas de comunicação em redes de computadores, compressão de arquivos, etc...
  - **software aplicativo**
    - programas desenvolvidos ou adquiridos pelos usuários para algum fim específico
    - de natureza profissional, educacional ou mesmo de lazer (jogos)

Programação de Computadores

- Linguagem de Programação
  - **Definida como sendo um**
    - conjunto limitado de **instruções** (vocabulário)
    - associado a um conjunto de regras (**sintaxe**)
      - define como as instruções podem ser associadas
      - como se pode compor os programas para a resolução de um determinado problema
- Existem várias linguagens de programação
  - **algumas de uso mais geral**
  - **outras concebidas para áreas de aplicação específicas**

Níveis de Linguagens de Programação

- Classificação das linguagens de programação
  - **Podem ser classificadas em níveis de linguagens**
    - Sendo que os níveis mais baixos são mais próximos da linguagem interpretada pelo processador e mais distante das linguagens naturais
  - **Níveis:**
    - Linguagem de Máquina
    - Linguagem Hexadecimal
    - Linguagem Assembly
    - Linguagem de Alto nível

Linguagem de Máquina

- Computador
  - **Corresponde basicamente a um conjunto de circuitos**
    - Sua operação é controlada através de programas escritos numa forma bastante primitiva
      - baseada no sistema binário de numeração tanto para a representação dos dados quanto das operações
- Linguagem de Máquina
  - **Forma básica (em linguagem binária) de representação dos programas**
  - **É a forma compreendida e executada pelo hardware do sistema**

Linguagem de Máquina

- Representação
  - **Instruções de linguagem de máquina são representadas por códigos na forma de palavras binárias cuja extensão pode variar de 8 a 64 bits**
    - 0100010100011101010101000010010101010001011110110 1…
  - **Contém instruções elementares, como transferência de dados da memória para registros internos da CPU, adição de valores, etc.**
    - 1011000000000001 significa colocar valor 1 no registro interno AL
    - 1111111011000000 incrementa de 1 o valor do registro AL
  - **Programação impraticável para escrita e leitura**

Linguagem Hexadecimal

- Linguagem Hexadecimal
  - **Simplifica a compreensão e a programação de computadores**
    - seqüência de bits é representada por números hexadecimais
  - **Notação em hexadecimal**
    - 1011000000000001_b = $B001_h$ significa colocar valor 1 em AL
    - 1111111011000000_b = $FEC0_h$ incrementa de 1 AL
  - **Programação de aplicações diretamente em linguagem de máquina é impraticável**
    - mesmo representada na notação hexadecimal

Linguagem Assembly

- Linguagem Assembly (linguagem de montagem)
  - **linguagem de máquina de cada processador é acompanhada de uma versão "legível" da linguagem de máquina**
- Uma linguagem simbólica
  - **Pois não é composta de números binários/hexadecimais**
  - **Utiliza palavras abreviadas, chamadas de mnemônicos, indicando a operação**
    - MOV R1, R2
      - mnemônico MOV (abreviação de MOVE)
      - dois registradores como parâmetros: R1 e R2
      - processador comanda o movimento do conteúdo de R2 para R1
      - equivalente a instrução Pascal R1:=R2
    - ADD R1, R2
      - mnemônico ADD (abreviação de ADDITION)
      - dois registradores como parâmetros: R1 e R2.
      - processador comanda a adição do conteúdo de R1 ao conteúdo de R2 e o resultado é armazenado em R1
      - equivalente a instrução Pascal R1:=R1+R2

Linguagem Assembly

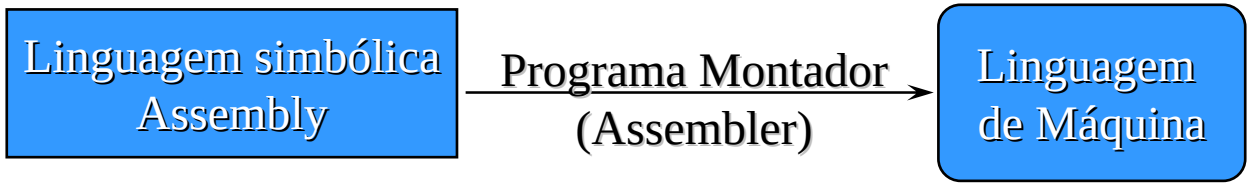
- Simplifica a programação em código de máquina
  - **Escolhendo nomes descritivos para as posições de memória e usando mnemônicos para representar códigos de operação**
  - **Exemplo: operação de dois números inteiros: v1:=v2+v3**
    - se associarmos o nome v2 à posição de memória 200_h, v3 à posição 202_h e v1 à posição 204_h
    - em notação hexadecimal ficaria: A1000203060202A30402
    - usando a técnica mnemônica:
      - MOV AX,v2 ; AX recebe o valor de memória associada a B
      - ADD AX,v3 ; AX recebe a soma de AX (B) com o valor de C
      - MOV v1,AX ; variável A recebe valor de AX

Linguagem Assembly

- Linguagem Assembly apresenta certas dificuldades
  - **Necessidade de definição de um conjunto relativamente grande de instruções para a realização de tarefas que seriam relativamente simples**
  - **Exigência do conhecimento de detalhes do hardware do sistema**
    - arquitetura interna do processador, endereços e modos de operação de dispositivos de hardware, etc...
- Tem vantagens
  - **Utilização da linguagem Assembly proporciona um maior controle sobre os recursos do computador**
    - permitindo também obter-se bons resultados em termos de otimização de código

Linguagem Assembly

- Geração do código de máquina
  - **Assembly é uma versão legível da linguagem de máquina**
    - passagem de um programa escrito em Assembly para a linguagem de máquina é quase sempre direta
      - não envolvendo muito processamento
  - **Passagem de um programa Assembly para linguagem de máquina é chamada de Montagem**
    - programa que realiza esta operação é chamado de **montador** (Assembler).



Linguagem Assembly

- Linguagem Assembly é orientada para máquina (processador)
  - **É necessário conhecer a estrutura do processador para poder programar em Assembly**
    - Utiliza instruções de baixo nível que operam com registros e memórias diretamente
      - é muito orientada às instruções que são diretamente executadas pelo processador
    - Não pode ser reutilizado em famílias de processadores diferentes
  - **Famílias geralmente mantém um certo nível de interoperabilidade**
    - Família x86 processador Pentium suporta o Assembly do 80486, que suporta o do 80386...
- Na seqüência da evolução
  - **Procurou-se aproximar mais a linguagem de programação à linguagem natural que utilizamos no dia-a-dia**
    - surgiram então as linguagens de alto nível, tipo Pascal, C, C++, etc

Linguagem Assembly

- Desvantagens com relação as linguagens de alto nível:
  - **Assembly apresenta um número muito reduzido de instruções**
    - do tipo operações de movimentação de dados em memória, para registros e para memórias, e operações lógicas e aritméticas bem simples
    - instruções assembly são de baixa expressividade
      - elas são de baixo nível
      - programador deve programar num nível de detalhamento muito maior para fazer a mesma coisa que em um programa escrito em linguagem de alto nível
  - **Programador utiliza diretamente os recursos do processador e memória**
    - ele deve conhecer muito bem a máquina onde ele está programando
  - **Programa escrito em linguagem Assembly não é muito legível**
    - por isso ele deve ser muito bem documentado

Linguagem Assembly

- Desvantagens com relação as linguagens de alto nível:
  - **Programa Assembly não é muito portável**
    - portável apenas dentro de uma família de processadores
  - **Assembly tem um custo de desenvolvimento maior**
    - devido a sua baixa expressividade, ilegibilidade e exigência do conhecimento sobre a máquina faz a programação
    - requerendo um maior número de homens/hora comparado com a programação utilizando linguagens de alto nível

Linguagem Assembly

- Vantagens com relação as linguagens de alto nível:
  - **Permite o acesso direto ao programa de máquina**
    - utilizando uma linguagem de alto nível não tem-se o controle do código de máquina gerado pelo compilador
    - programador pode gerar um programa mais compacto e eficiente que o código gerado pelo compilador
      - pode ser 0 ou 300 % menor e mais rápido que um programa compilado
  - **Assembly permite o controle total do hardware**
    - por exemplo, permitindo a programação de portas seriais e paralela de um PC

Linguagem Assembly

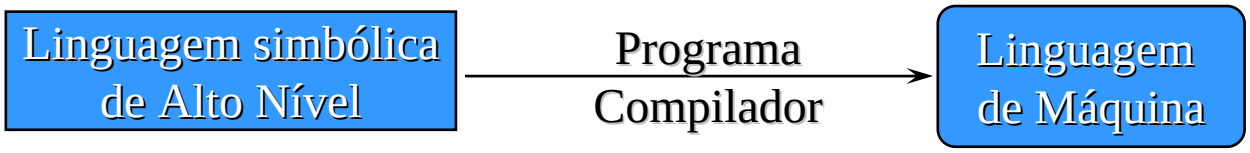
- Aplicações da Linguagem Assembly
  - **Controle de processos com resposta em tempo real**
    - devido a possibilidade de gerar programas mais eficientes
    - Aplicação tempo-real
      - processador deve executar um conjunto de instruções em um tempo limitado
        - » exemplo: a cada 10 ms processador deve ler um dado, processá-lo e emitir um resultado
  - **Comunicação / transferência de dados**
    - devido a possibilidade de acessar diretamente o hardware
    - linguagem Assembly é utilizada para a implementação de programas de comunicação ou transferência de dados
  - **Otimização de sub-tarefas da programação de alto nível**
    - um programa não precisa somente ser escrito em linguagem Assembly ou linguagem de alto nível
    - existem programas de alto nível com sub-tarefas escritas em linguagem Assembly
      - pode-se otimizar partes de programas, no caso de tarefas tempo-real
      - para a programação do hardware do computador

Linguagem de Alto Nível

- Linguagem de Alto Nível
  - **Assim denominadas por apresentarem uma sintaxe mais próxima da linguagem natural**
  - **Fazem uso de palavras reservadas extraídas do vocabulário corrente (como READ, WRITE, TYPE, etc...)**
  - **Permitem a manipulação dos dados nas mais diversas formas**
    - números inteiros, reais, vetores, listas, etc...
    - enquanto a linguagem Assembly trabalha com bits, bytes, palavras, armazenados em memória.
- Origem e Exemplos
  - **Surgiram entre o final da década de 50 e início dos anos 60**
    - Fortran, Cobol, Algol e Basic
  - **Algumas delas têm resistido ao tempo e às críticas**
    - Fortran ainda é visto como uma linguagem de implementação para muitas aplicações de engenharia
    - Cobol é uma linguagem bastante utilizada no desenvolvimento de aplicações comerciais

Linguagem de Alto Nível

- Linguagem de Alto Nível para linguagem de máquina
  - **Passagem é bem mais complexa**
    - são utilizados compiladores e linkadores



Linguagem de Alto Nível

- Permitiria Interoperabilidade
  - **Dado que os comandos das linguagens de alto nível não referenciam os atributos de uma dada máquina**
    - podem ser facilmente compilados tanto em uma máquina como em outra
  - **Programa escrito em linguagem de alto nível poderia, teoricamente, ser usado em qualquer máquina**
    - bastando escolher o compilador correspondente
- Interoperabilidade não é tão simples
  - **Quando um compilador é projetado**
    - certas restrições impostas pela máquina são refletidas como características da linguagem a ser traduzida
    - Exemplo
      - tamanho do registrador e as células de memória de uma máquina limitam o tamanho máximo dos inteiros que nela podem ser convenientemente manipulados
  - **Em diferentes máquinas uma "mesma" linguagem pode apresentar diferentes características, ou dialetos**
    - é necessário fazer ao menos pequenas modificações no programa antes de move-lo de uma máquina para outra

Linguagem de Alto Nível

- Problema da Interoperabilidade
  - **Para auxiliar a interoperabilidade**
    - American National Standards Institute (ANSI) e a International Organization for Standardization (ISO) adotaram e publicaram padrões para muitas das linguagens mais populares
    - também surgiram padrões informais
      - devido à popularidade de um dados dialeto de uma linguagem e ao desejo oferecerem produtos compatíveis

Linguagens de Alto Nível

- Classificação quanto ao uso
  - Linguagens de uso geral
    - podem ser utilizadas para implementação de programas com as mais diversas características e independente da área de aplicação
    - Pascal, Modula-2 e C;
  - Linguagens especializadas
    - são orientadas ao desenvolvimento de aplicações específicas
    - Prolog, Lisp e Forth;
  - Linguagens orientadas a objeto
    - oferecem mecanismos sintáticos e semânticos de suporte aos conceitos da programação orientada a objetos
    - Smalltalk, Eiffel, C++ e Delphi

Desenvolvimento de Programas

- Ambiente de desenvolvimento
  - **Desenvolvimento de programas é associado ao uso de ferramentas ou ambientes de desenvolvimento**
    - que acompanham o programador desde a etapa de codificação propriamente dita até a geração e teste do código executável
  - **Principais etapas de geração de um programa**
    - Codificação do código fonte
    - Tradução do código fonte
    - Linkagem
    - Depuração
  - **Engenharia de Software**
    - Metodologias mais completas que definem os passos para o desenvolvimento de programas

Desenvolvimento de Programas

- Tradução do Código Fonte (código objeto)
  - **Compreende três atividades: análise léxica, análise sintática e geração de código**
    - são processadas pelos módulos do tradutor: analisadores léxico, sintático e gerador de código
  - **Análise léxica**
    - É o processo de reconhecer quais cadeias de símbolos do programa-fonte representam entidades indivisíveis
    - Exemplos
      - três símbolos 153 não devem ser interpretados como 1, seguido por 5, seguido por 3, mas são reconhecidos como um valor numérico apenas
      - palavras que aparecem no programa, embora compostas de caracteres individuas, devem ser interpretadas cada qual como uma unidade inseparável

Desenvolvimento de Programas

- Tradução do Código Fonte (código objeto)
  - **Análise léxica**
    - Analisador léxico identifica um grupo de símbolos que representam uma única entidade
      - classifica como sendo um valor numérico, ou uma palavra, ou um operador aritmético, e assim por diante
      - gera um padrão de bits conhecido como átomo (token), indicativo da classe do elemento
    - Átomos são os dados de entrada do analisador sintático

Desenvolvimento de Programas

- Tradução do Código Fonte (código objeto)
  - **Análise sintática**
    - Processo de identificação da estrutura gramatical do programa, e de reconhecimento do papel de cada um dos seus componentes
    - Processo de análise sintática é feito com base em um conjunto de regras sintáticas que definem a sintaxe da linguagem de programação
    - Uma forma de expressar regras sintáticas é através de diagramas de sintaxe

Desenvolvimento de Programas

- Tradução do Código Fonte (código objeto)
  - **Árvore sintática**
    - Processo de análise sintática de um programa consiste em construir uma árvore de sintaxe para o programa-fonte
    - Por isso as regras de sintaxe que descrevem a estrutura gramatical de um programa não devem propiciar que duas ou mais árvores de sintaxe distintas possam ser construídas para a mesma cadeia

Desenvolvimento de Programas

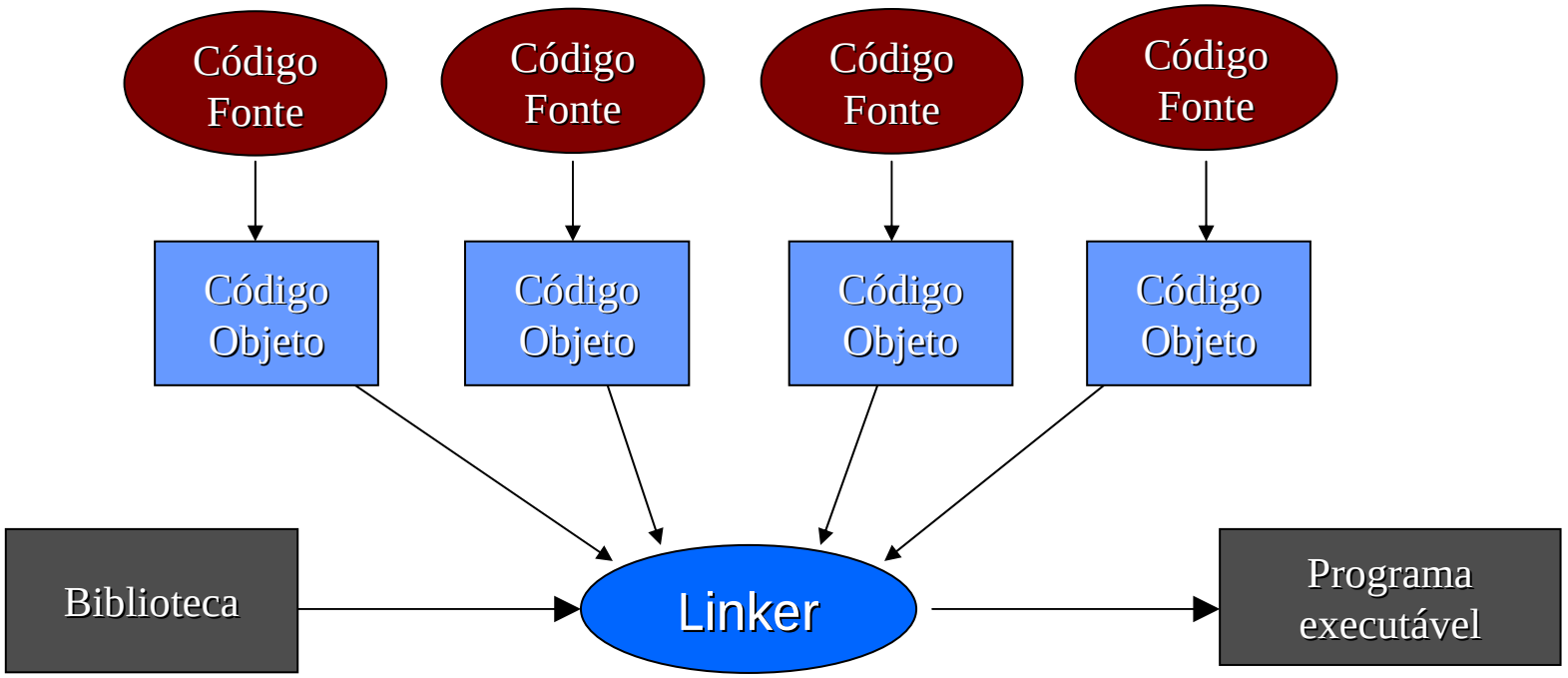
- Tradução do Código Fonte (código objeto)
  - **À medida que um analisador sintático recebe átomos (partes do código) do analisador léxico, ele vai analisando os comandos e ignorando os comentários**
  - **As informações extraídas das declarações são tabeladas em uma estrutura conhecida como tabela de símbolos**
    - guarda informações sobre as variáveis declaradas, os tipos de dados e as estruturas de dados associadas a tais variáveis

Desenvolvimento de Programas

- Tradução do Código Fonte (código objeto)
  - **Análises léxica e sintática e a geração de código não são efetuadas em ordem estritamente seqüencial, mas de forma intercalada**
  - **Analisador léxico começa identificando o primeiro átomo e fornecendo-o ao analisador sintático**
    - com esta pista sobre a estrutura que vem a seguir, o analisador sintático solicita ao analisador léxico o próximo átomo
  - **À medida que o analisador sintático reconhece sentenças ou comandos completos**
    - vai ativando o gerador de código
    - para que este possa produzir as correspondentes instruções de máquina

Desenvolvimento de Programas

- Editores de ligação (linker)
  - **Objetivo: rearranjar o código do programa incorporando a ele todas as partes referenciadas no código original**
    - resultando num código executável pelo processador
  - **Tarefa é feita pelos ligadores (linkadores)**



Desenvolvimento de Programas

- Depuradores ou debuggers
  - **Auxilia o programador a eliminar (ou reduzir) a quantidade de “bugs” (erros) de execução no seu programa**
  - **Executam o programa gerado através de uma interface apropriada que possibilita uma análise efetiva do código do programa**
  - **Graças à:**
    - execução passo-a-passo (ou instrução por instrução) de partes do programa
    - visualização do “estado” do programa através das variáveis e eventualmente dos conteúdos dos registros internos do processador
    - alteração em tempo de execução de conteúdos de memória ou de variáveis ou de instruções do programa
    - etc...

Paradigmas de programação

- Existem vários paradigmas de programação
  - **que são estilos utilizados pelos programadores para conceber um programa**
  - **os mais conhecidos são**
    - Programação não-estruturada
    - Programação Procedural
    - Programação Modular
    - Programação Orientada a Objetos

Paradigmas de programação

- Programação não-estruturada
  - **Pessoas aprendem a programação escrevendo programas pequenos e simples**
    - consistindo apenas de um programa principal
    - uma seqüência de comandos ou declarações que modificam dados que são acessível a todos os pontos do programa
  - **Técnica de programação não estruturada**
    - tem várias desvantagens no caso de programas grandes
      - se a mesma seqüência é necessária em localizações diferentes ela deve ser copiada
      - não permite a organização do programa

Paradigmas de programação

- Programação Procedural
  - **Programa é visto como uma seqüência de chamadas de procedimentos**
  - **Uma chamada de procedimento é usado para invocar o procedimento**
    - podendo ser passado alguns parâmetros
  - **Após a seqüência ser executada**
    - controle retorna justo após o ponto de chamada do procedimento
  - **Programas podem ser escritos mais estruturados e livres de erro**
    - introduzindo parâmetros tão bem quanto procedimentos de procedimentos (sub-procedimentos)

Paradigmas de programação

- Programação Modular
  - **No passar dos anos**
    - ênfase no projeto de programas passou do projeto de procedimentos para a organização dos dados
      - surgindo a programação modular
  - **Procedimentos relacionados e dados que eles utilizam são agrupados em módulos separados**
    - por exemplo, todas as funções de manipulação de uma pilha (empilhar, desempilhar, etc.) e a pilha em si podem ser agrupadas em um módulo
  - **Cada módulo tem seus próprios dados**
    - permite que cada módulo gerencie um estado interno que é modificado por chamadas a procedimentos deste módulo

Paradigmas de programação

- Programação Orientada a Objetos
  - **Temos uma malha de objetos que interagem**
    - cada um mantendo seu próprio estado
  - **Essência da programação orientada a objetos**
    - consiste em tratar os dados e os procedimentos que atuam sobre os dados como um único objeto
  - **Objeto**
    - entidade independente com uma identidade e certas características próprias

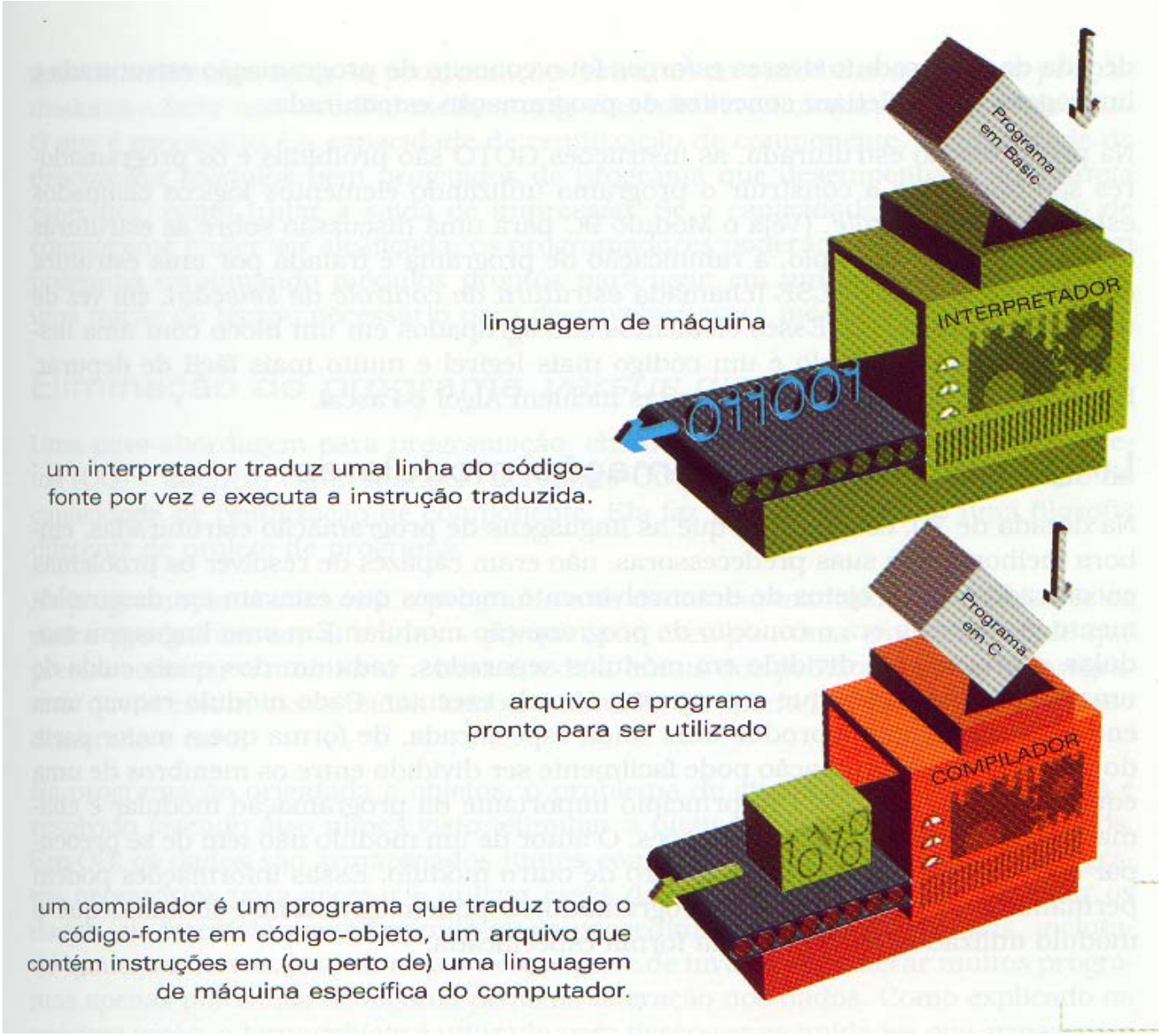
Paradigmas de programação

- Linguagens de programação e seus paradigmas
  - **Uma linguagem de programação fornece o suporte a um estilo ou paradigma de programação se ela fornece funcionalidades que a tornam conveniente para usar determinado estilo**
  - **Uma linguagem não suporta uma técnica se é necessário esforços excepcionais ou destreza para escrever tal programa**
    - ela meramente habilita a técnica a ser usada
    - Por exemplo, pode-se escrever programas estruturados em Fortran e programas orientados a objetos em C
      - mas isto é desnecessariamente difícil de fazer porque estas linguagens não suportam diretamente estas técnicas
  - **É de responsabilidade do programador aplicar certa técnica de programação**

Linguagens Interpretadas

- Linguagens de programação compiladas
  - **Etapas de desenvolvimento anteriores consideram que o uso de linguagens de programação compiladas**
    - aquelas que produzirão um programa na forma da linguagem de máquina do processador
    - instruções definidas pelo programador usando uma linguagem de alto nível serão traduzidas para as instrução na linguagens de máquina
- Linguagens de programação interpretadas
  - **Interpretação do programa usando outro programa, chamado interpretador**
  - **Linguagem interpretada mais conhecida: a linguagem Java**
    - interpretada por uma máquina virtual chamada JVM (Java Virtual Machine)
- Interpretador
  - **Programa que executa as instruções escritas em linguagem de alto nível**
    - Geralmente translada as instruções de alto nível em uma forma intermediária que é executada

Linguagens Interpretadas



Linguagens Interpretadas

- Compilador Versus Interpretador
  - **Programa compilado executa mais rapidamente que um programa interpretado**
  - **Vantagem do interpretador é que ele não necessita passar por um estágio de compilação durante a qual as instruções de máquina são gerados**
    - Processo de tradução pode consumir muito tempo se o programa é longo
    - Interpretador pode executar imediatamente os programas de alto-nível
  - **Interpretadores são algumas vezes usados durante o desenvolvimento de um programa**
    - quando um programador deseja testar rapidamente seu programa
  - **Interpretadores são com freqüência usados na educação**
    - pois eles permitem que o estudante programe interativamente.

Linguagens Interpretadas

- Compilador Versus Interpretador
  - **Tanto os interpretadores como os compiladores são disponíveis para muitas linguagens de alto nível**
    - Java, Basic e LISP são especialmente projetadas para serem executadas por um interpretador
  - **Linguagens interpretadas podem possibilitar a portabilidade**
    - Como não é gerado um código de máquina, e sim um código intermediário que será interpretado por uma máquina virtual
    - Pode-se obter a portabilidade do código se esta máquina virtual for desenvolvida para várias plataformas
      - este é o caso da linguagem Java

Linguagens Interpretadas

- Máquina Virtual
  - **É um ambiente operacional**
    - ambiente onde os usuários executam o programa
    - auto-contido que se comporta como se fosse um computador separado
  - **Exemplo: applet Java executa em uma Máquina Virtual Java que não acessa ao sistema operacional do computador hospedeiro**
  - **Este projeto tem duas vantagens:**
    - Independência de sistema
      - uma aplicação poderá ser executada em qualquer máquina virtual, sem se preocupar com o hardware e software dando suporte ao sistema
    - Segurança
      - como a máquina virtual não tem contato com o sistema operacional, existem poucas possibilidade de um programa interpretado danifique outros arquivos ou aplicações
      - Esta vantagem trás consigo uma limitação: os programas executando em uma máquina virtual não pode tomar vantagem das funcionalidades do sistema operacional