Maurício Dourado
mauriciodourado@ifce.edu.br

Paradigmas de Programação

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CEARÁ

Aula 02 – Apresentação

O que vimos?

- Conceito de Paradigma
- Tipos de Paradigmas
- História
- Paradigma x Linguagem de Programação

Hoje

- Fatores que influenciam o projeto de linguagem de programação
- Critérios de Avaliação de Linguagem de Programação

FATORES QUE INFLUENCIAM O PROJETO DE LP

Características

- Legibilidade (Readability)
  - Grau de facilidade de ler e entender um programa
- Capacidade escrita (Writability)
  - Grau de facilidade de usar uma linguagem para criar programas
- Confiabilidade (Reliability)
  - Conformidade com as especificações de acordo as condições impostas
- Custo
  - O custo final é um dos principais elementos na avaliação de qualquer linguagem de programação

Legibilidade (*Readability*)

- Ortogonalidade
  - Um conjunto relativamente pequeno de construções primitivas pode ser combinado em um número relativamente pequeno de maneiras
  - Toda combinação possível é legal e significativa

Legibilidade (*Readability*) Ortogonalidade

Um exemplo: adicionar valores inteiros de 32 bits que residem na memória ou nos registradores e substituir um dos valores pela soma.

Computadores de grande porte da IBM

```
A    Reg1, célula_de_memória
AR Reg1, Reg2
```

Semântica:

```
Reg1 ← conteúdo(Reg1) + conteúdo(célula_de_memória)
Reg1 ← conteúdo(Reg1) + conteúdo(Reg2)
```

Instrução de adição do VAX

```
ADDL operando_1, operando_2
```

Semântica:

```
operando_2 ← conteúdo(operando_1) + conteúdo(operando_2)
```

O projeto da instrução VAX é ortogonal, pois uma única operação pode usar registradores ou células de memória como operandos.

Legibilidade (*Readability*) Ortogonalidade

- Instruções de Controle
  - Existência de estruturas de controle bem conhecidas

```
while (incr < 20) {
      while (sum <= 100) {
       sum += incr;
      }
      incr++;
}
```

```
loop1:
      if (incr >= 20) goto out;
loop2:
      if (sum > 100) goto next;
            sum += incr;
            goto loop2;
next:
      incr++;
      goto loop1;
out:
```

Legibilidade (*Readability*)

- Tipos de dados e estruturas
  - A presença de facilidades adequadas para definir tipos de dados e estruturas de dados
  - Exemplo: suponha que em uma linguagem não exista um tipo de dado booleano e um tipo numérico seja usado para substituí-lo:
    - timeOut = 1 (significado não claro)
    - timeOut = true (significado claro)

Legibilidade (*Readability*)

- Considerações sobre a sintaxe
  - Formas identificadoras
    - Restringir os identificadores a tamanhos muito pequenos prejudica a legibilidade
  - Palavras especiais
    - Formas das palavras especiais de uma linguagem (exemplo: while, class, for e begin-end)
    - Palavras especiais de uma linguagem podem ser usadas como nomes de variáveis?
  - Forma e significado
    - Projetar instruções de forma que sua aparência indique sua finalidade

Capacidade de Escrita (*Writability*)

- Simplicidade e ortogonalidade
  - Poucos construtores, um pequeno número de primitivas, um pequeno conjunto de regras para combiná-los
- Suporte para abstração
  - A capacidade de definir e de usar estruturas ou operações complexas de maneira que permita ignorar muitos dos detalhes
- Expressividade
  - Um conjunto relativamente conveniente de maneiras de especificar operadores
  - Exemplos:
    - count++ é mais conveniente que count = count + 1
    - a inclusão do for em muitas linguagens modernas

Confiabilidade (*Reliability*)

- Verificação de tipos
  - Testar se existem erros de tipos
- Manipulação de Exceções
  - Capacidade de interceptar erros em tempo de execução e por em prática medidas corretivas
- Apelidos (Aliasing)
  - Presença de dois ou mais métodos, ou nomes, distintos que referenciam a mesma célula de memória
- Legibilidade (Readability) e Capacidade de Escrita (Writability)
  - Uma linguagem que não suporta maneiras naturais de expressar os algoritmos usará, necessariamente, abordagens não-naturais. Assim, a legibilidade será reduzida
  - A legibilidade afeta a confiabilidade tanto na escrita quanto na manutenção

Custo

- Treinamento dos programadores para usar a linguagem
- Escrita de programas na linguagem
- Compilação programas na linguagem
- Execução dos programas
- Sistema de implementação da linguagem:
  - existência de compiladores free
- Confiabilidade
  - Confiabilidade baixa leva a altos custos ° Manutenção dos programas

Custo

- Custo de execução de programas:
  - otimização de compiladores, alocação de registros eficiente, mecanismos eficientes de suporte à run-time.
    - Foi um fator muito importante até a metade dos anos 90.
      - Início dos anos 90: XT com dois drivers, um para SO e outro para a linguagem de programação e as aplicações.
    - Hoje, entretanto, para muitas aplicações, velocidade de execução não é mais a principal preocupação.
      - Computadores atuais rodam milhões de instruções por segundo.

Custo

- Custo de criação, teste e uso de programas:
  - O esforço gasto para resolver um problema através da implementação de uma aplicação deve ser minimizado.
    - A linguagem de programação deve prover ferramentas que facilitem estas tarefas
      - (1) Ambiente gráfico para desenvolvimento;
      - (2) composição (componetnes) ao invés de implementação;
      - (3) Ferramentas de debug;
      - (4) Automação de testes;
      - (5) Gerenciadores de versões, ...

Custo

- Custo de manutenção de programas:
  - Estudos confirmam que o tempo gasto com a manutenção de software é maior do que o tempo gasto com o seu desenvolvimento.
    - Manutenção inclui reparos de erros, mudanças nos requisitos originais, inserção de novos requisitos (novas demandas de mercado)
    - Linguagens de programação devem facilitar a manutenção de software.
      - Possibilitando um baixo acoplamento (efeito gelatina)
      - Uma tecnologia que facilita a manutenção: Frameworks e componentes.
        - » Encapsula o que há de comum em frameworks e as particularidades em componentes.
        - » Permite um baixo acoplamento.
        - » Possibilita a criação e inserção de novos componentes de acordo com a demanda.

Custo

Sabendo que esses são funções da capacidade de escrita e da legibilidade.

Outros Critérios de Avaliação

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

Custo/Benefício no projeto da linguagem

- Confiabilidade versus Custo de Execução
  - Critérios conflitantes
  - Exemplo: Java requer que todas as referências a vetores sejam checadas para garantir que os índices estejam dentro dos limites, mas isso aumenta o custo de execução
- Readability versus writability
  - Critérios conflitantes
  - 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
- Writability (flexibility) versus reliability
  - Critérios conflitantes
  - Exemplo: ponteiros em C++ são poderosos e muito flexíveis

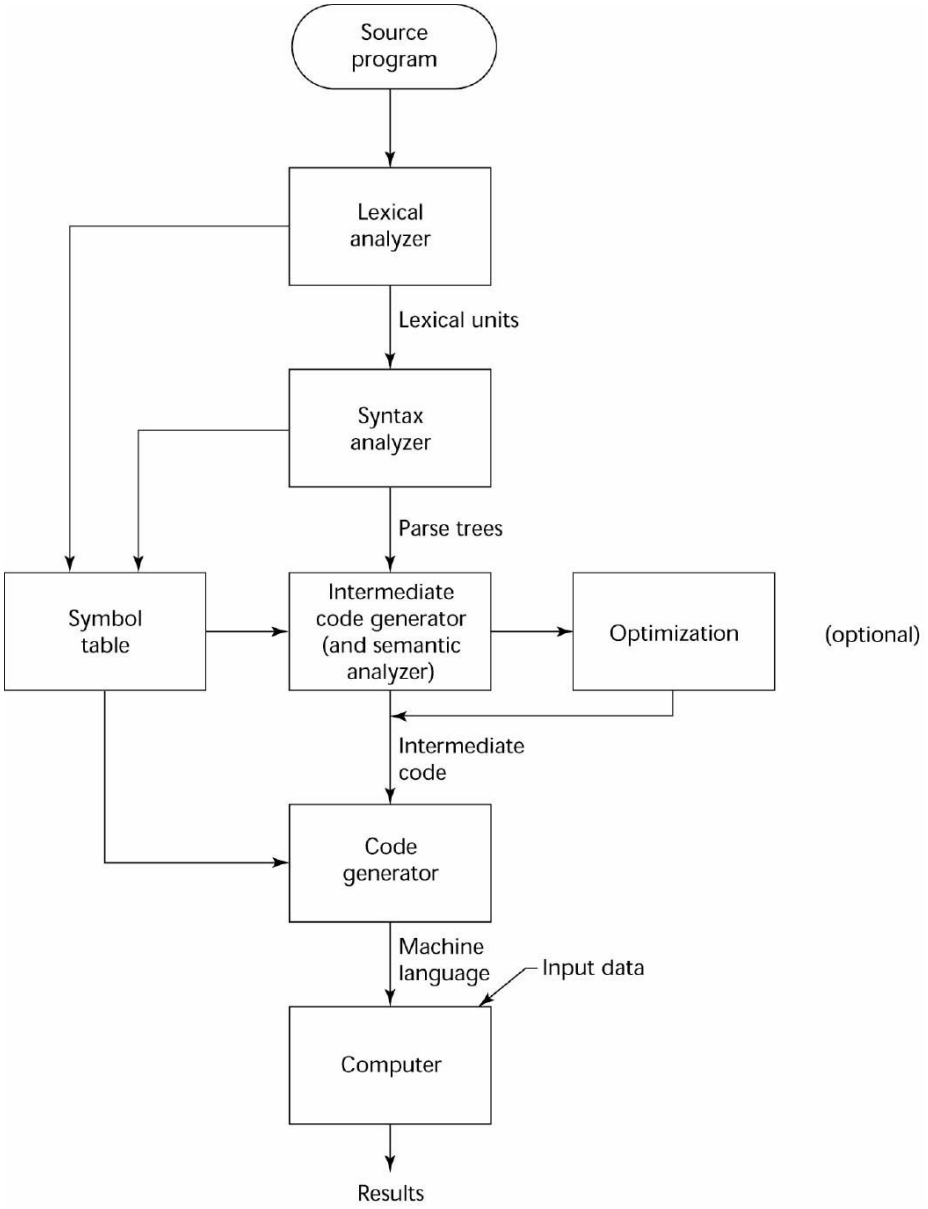
Métodos de Implementação

- Compilação
  - Programas são traduzidos para linguagem de máquina
- Interpretação pura
  - Programas são interpretados por outro programa conhecido como interpretador
- Sistemas de Implementação Híbridos
  - Um meio-termo entre compiladores e interpretadores puros

Compilação

- Traduz programas em alto-nível (linguagem fonte) em código de máquina (linguagem de máquina)
- Tradução lenta, execução rápida
- O processo de compilação possui várias fases:
  - Análise léxica
    - Converte caracteres de um programa fonte em unidades léxicas
  - Análise sintática
    - Transforma unidades léxicas em parse trees, as quais representam a estrutura sintática do programa
  - Análise semântica
    - Gera código intermediário
  - Geração de código
    - Código de máquina é gerado

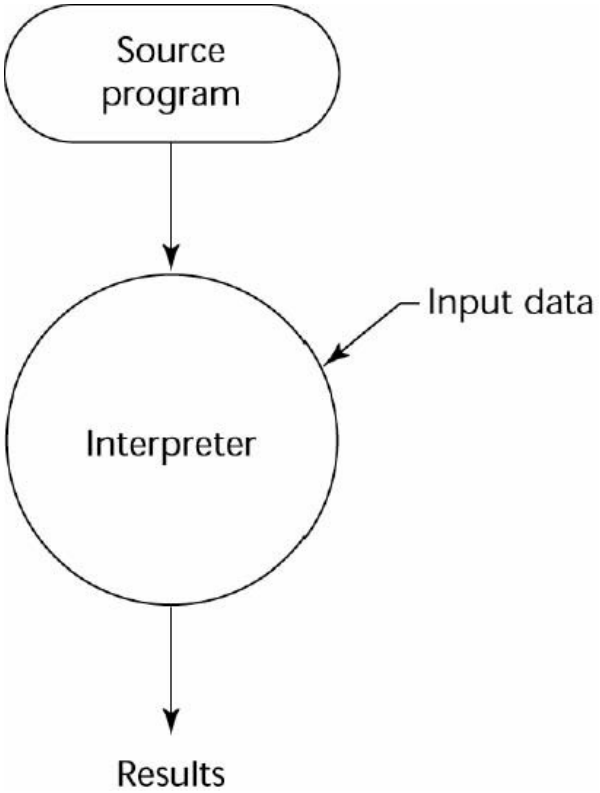
O Processo de Compilação



Interpretação Pura

- Sem tradução
- Fácil implementação de programas (erros de execução podem ser facilmente e rapidamente mostrados)
- Execução lenta (de 10 a 100 vezes mais lenta do que programas compilados)
- Geralmente requer mais espaço
- Cada vez mais raro em linguagens de alto-nível

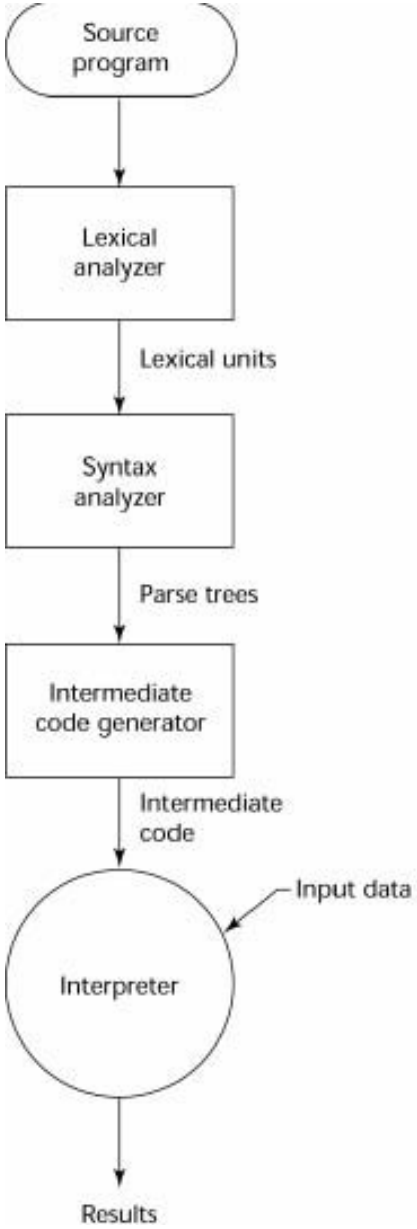
O Processo de Interpretação Pura



Sistemas de Implementação Híbridos

- Um meio-termo entre compilador e interpretador puro
- Um programa em uma linguagem de alto-nível é traduzido para uma linguagem intermediária que permite fácil interpretação
- Mais rápido do que interpretação pura
  - Exemplos
    - Programas em Perl são parcialmente compilados para detectar erros antes da interpretação
    - Java
      - Byte codes, permitem portabilidade para qualquer máquina que possui a Java Virtual Machine

O Processo de Implementação Híbrida



Bibliografia

- Página 22 a 40 do livro texto