



Plano de Ensino


- Revisão de Conjuntos e Funções
- Linguagens, Expressões Regulares e Gramáticas
- Autômatos
- **Conceitos básicos sobre compiladores e interpretadores**
- **Visão geral do processo de compilação**
- **Tipos de compiladores**
- Análise léxica
- Análise sintática
- Análise semântica
- Geração de Código





Livro-Texto

- **Bibliografia Básica:**
 - » AHO, A.; ULLMANN, J.; REVI, S.. Compiladores : princípios, técnicas e ferramentas. 3ª ed. Rio de Janeiro: LTC, 2006.
- **Bibliografia Complementar:**
 - » TOSCANI, Simão Siríneo; PRICE, Ana M. A.. Implementação de Linguagens de Programação. 1ª ed. Porto Alegre: Bookman Companhia Ed., 2008.
 - » DELAMARO, Marcio Eduardo. Como Construir um Compilador : Utilizando Ferramentas Java. 1ª ed.: Novatec, 2004.



7. Compiladores e Interpretadores



- Um tradutor é um programa que recebe como dado de entrada um programa escrito em uma linguagem de programação (**a linguagem fonte**) e produz como saída de seu processamento um programa escrito em outra linguagem (**a linguagem objeto**).
- Se a linguagem fonte é uma linguagem de alto nível como Pascal ou C, e a linguagem objeto é uma linguagem de baixo nível como a linguagem de montagem ("assembly") ou de máquina, o tradutor é chamado de compilador.

7. Compiladores e Interpretadores



- Por esse enfoque, a execução de um programa escrito em linguagem de programação de alto nível é basicamente um processo de dois passos:
 - » O programa fonte deve primeiro ser compilado, isto é, traduzido para a linguagem objeto
 - » Após compilado o programa objeto é então carregado na memória e executado.

7. Compiladores e Interpretadores



1ª Fase → Compilação



2ª Fase → Execução



7. Compiladores e Interpretadores



- **Interpretador** → certos tipos de tradutores transformam uma linguagem de programação (LP) em uma linguagem simplificada, chamada de código intermediário, que pode ser diretamente “executado” por um programa chamado interpretador.
- Podemos imaginar o código intermediário como uma linguagem de máquina de um computador abstrato projetado para executar este tipo de código intermediário.
 - » **Exemplo:** Prolog e Java, são linguagens interpretadas.

7. Compiladores e Interpretadores



- Interpretadores são, em geral, menores que compiladores e facilitam a implementação de construções complexas em LPs.
- Entretanto, o tempo de execução de um programa interpretado é geralmente maior que o tempo de execução desse mesmo programa compilado.

7. Compiladores e Interpretadores



- **Montadores (“Assemblers”)** → traduzem programas escritos em linguagem de montagem nos correspondentes programas escritos linguagem de máquina (0’s e 1’s).
- **Pré-processadores** → traduzem programas escritos em linguagens de alto nível em outros programas escritos também em linguagens de alto nível (exemplo: o pré-processador da linguagem C).

7. Compiladores e Interpretadores



- **Macroprocessadores** → semelhantes aos pré-processadores, traduzem programas escritos em linguagens de alto nível em outros programas também escritos em linguagens de alto nível, tendo também a capacidade de processamento de macro-instruções.
- Uma macro-instrução é um nome simbólico associado a um conjunto de instruções de uma LP que pode ser usado como referência a esse conjunto de instruções dentro do programa. Uma macro-instrução pode ou não ter parâmetros.

7. Compiladores e Interpretadores



» Exemplo:

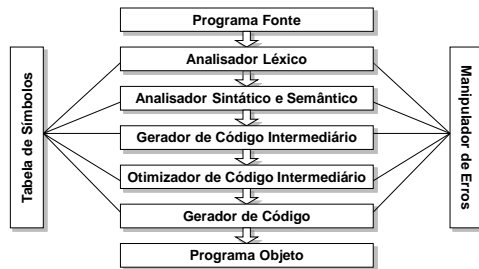
```
#define MIN(A,B) ((A) <= (B) ? (A) : (B))  
...  
int x, y, z;  
...  
z = MIN(x,y);
```

7. Compiladores e Interpretadores



- **Compilador** → recebe como entrada um programa fonte e produz como saída um programa objeto na forma de um conjunto de instruções em linguagem de máquina (ou, mais comum hoje em dia, em linguagem de montagem).
- Pela complexidade do processo, divide-se o mesmo em uma série de subprocessos chamadas fases. Uma fase é uma operação que toma como entrada uma representação do programa fonte e produz como saída uma outra representação.

7. Compiladores e Interpretadores



7. Compiladores e Interpretadores



- **Analizador Léxico (Scanner)** → lê o programa fonte caractere a caractere, agrupando a sequência de caracteres lidos em grupos de símbolos (*tokens*).
 - » Os símbolos são as palavras-chaves tais como IF e FOR.
 - » Identificadores de variáveis e procedimentos tais como X e SOMA.
 - » Operadores tais como + e <=.
 - » Símbolos de pontuação como parênteses e ponto-e-vírgula.
- A saída do analisador léxico é uma sequência de símbolos que é passada para a próxima fase, o analisador sintático.

7. Compiladores e Interpretadores



- **Analizador Sintático (Parser)** → agrupa os símbolos recebidos do analisador léxico em estruturas sintáticas.
 - » Os três símbolos representando A + B poderiam ser agrupadas em uma estrutura sintática chamada expressão.
 - » Expressões poderiam ser posteriormente agrupadas para formar comandos, e assim por diante.

7. Compiladores e Interpretadores



- **Analisador Semântico** → verifica se expressões geradas pelo analisador sintático, embora corretas sintaticamente, têm significado admissível na linguagem.
 - » Por exemplo, $A + B$ pode ser uma expressão sintaticamente correta, mas pode não ter significado em muitas linguagens, caso A seja inteiro e B seja um caractere.

7. Compiladores e Interpretadores



- **Gerador de Código** → usa as estruturas produzidas pelo analisador sintático e verificadas pelo analisador semântico para criar uma sequência de instruções simples dita código intermediário
 - » Este código intermediário está entre a linguagem de alto nível e a linguagem de baixo nível.

7. Compiladores e Interpretadores



- **Otimizador de Código** → é um módulo opcional, independente de máquina e presente na grande maioria dos compiladores, que objetiva melhorar o código intermediário de modo que o programa objeto produzido ao fim da compilação seja menor.
 - » Garante menor espaço de memória.
 - » Execução mais rápida
- Sua saída é um novo código intermediário.

7. Compiladores e Interpretadores



▪ **Gerador de Código** → produz o código objeto final, tomando decisões com relação à alocação de espaço para os dados do programa, selecionando a forma de acessá-los, definindo que registradores da CPU serão usados, etc.

» Projetar um gerador de código que produza programas objeto verdadeiramente eficientes é uma das tarefas mais difíceis no projeto de um compilador.

7. Compiladores e Interpretadores



▪ **Tabela de Símbolos** → este módulo de gerência de tabela de símbolos tem por função guardar informações a respeito de todos os nomes usados pelo programa e registrar informações importantes associadas a cada um, tais como seu tipo (inteiro, real, etc.), tamanho, escopo, etc.

7. Compiladores e Interpretadores



▪ **Manipulador de Erros** → é ativado sempre que for detectado um erro no programa fonte. Ele deve avisar o programador da ocorrência do erro emitindo uma mensagem, e ajustar-se novamente à informação que está sendo passada de fase a fase de modo a poder completar o processo de compilação.

» Mesmo que não seja mais possível gerar código objeto, a análise léxica e sintática deve prosseguir até o fim.

7. Compiladores e Interpretadores



- **Passo** → é a combinação de fases da compilação em bloco.
 - » Um passo lê o programa fonte ou a saída gerada pelo passo anterior, fazendo as transformações específicas de suas fases, gravando sua saída em um arquivo temporário que vai ser lido pelo passo posterior.
 - » Quando várias fases são agrupadas em um único passo, suas operações podem ser intercaladas com o devido controle de alternância entre elas.
 - » O número de passos e o agrupamento de fases em cada passo são geralmente definidos em função da estrutura da LP e das características da arquitetura para a qual o compilador vai gerar código.



Compiladores

clayton.valdo@anhanguera.com