Disciplina:

Programação Computacional

Prof. Fernando Rodrigues
e-m@il: fernandorodrigues@sobral.ufc.br

Aula 05_B: Introdução aos Algoritmos:

- Explanação das etapas de construção de um algoritmo;
- Explanação das fases de um processo de compilação;
- Diferença entre compilador e interpretador;
- Conceito de eficiência de um algoritmo.

Etapas de construção de um algoritmo

Problema: Identificar o problema é o primeiro passo no processo de construção de algoritmo;

Análise: Entender o problema é primordial para a resolução do mesmo.

Desenvolvimento da solução: Desenvolvimento do algoritmo;

Testes: Executar o algoritmo com dados conhecidos para obter um resultado esperado;

Alterações: Realizar alterações buscando sempre a velocidade e qualidade;

Algoritmo Final;

Processo de compilação

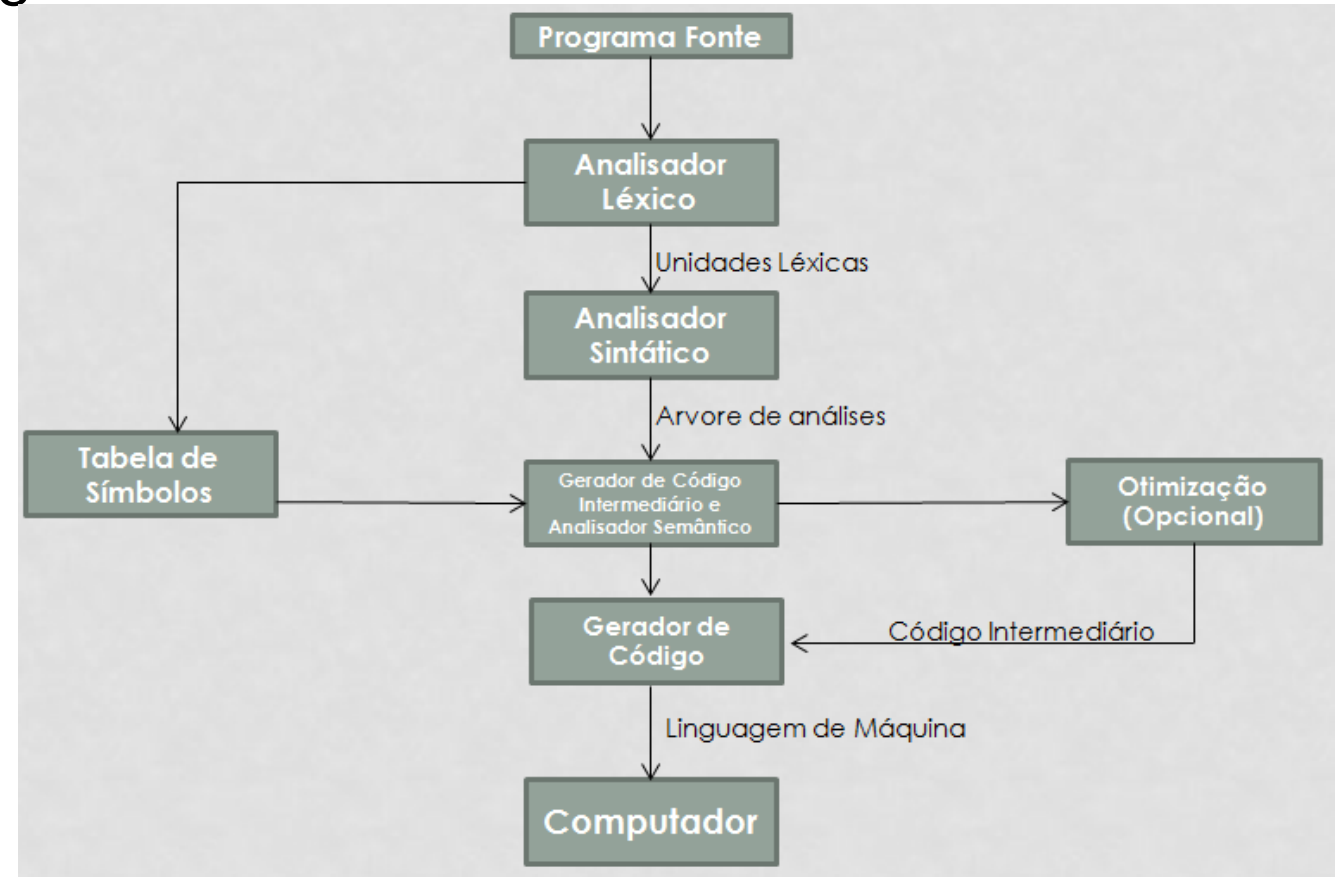
- Compilação é o processo de tradução de um código fonte, escrito normalmente em uma linguagem de alto nível (de fácil entendimento por parte do programador), para uma linguagem de baixo nível (em geral, linguagem de máquina).

- Um compilador é um programa responsável por executar os processos de compilação (descritos a seguir).

- Cada linguagem tem um compilador específico para cada plataforma (Sistema operacional) a qual a mesma se destina.

Processo de compilação

- Linguagens convencionais são compiladas para código nativo

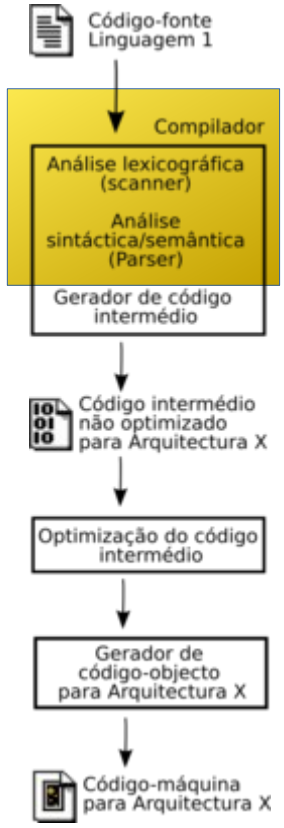


Fases da compilação (I)

- Análise Léxica
  - Esta é a primeira etapa da compilação. A função do analisador léxico, também conhecido como scanner, é analisar todo o código fonte e produzir símbolos (tokens) que podem ser manipulados na etapa seguinte. Nesta etapa são eliminados os espaços em branco e comentários.

- Análise Sintática
  - O analisador sintático (parsing) é quem dá significado às sequências de tokens criadas anteriormente.

- Análise Semântica
  - Esta etapa é responsável por analisar a semântica, ou significado, de cada elemento do código. É ela quem encontra erros como, por exemplo, uma multiplicação entre tipos de dados diferentes.

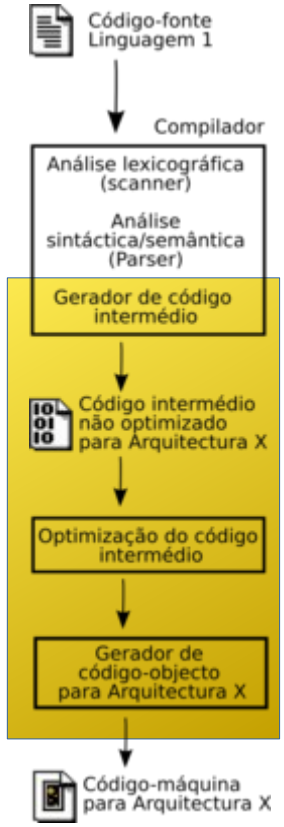


Fases da compilação (II)

- Geração do Código Intermediário
  - Nesta etapa ocorre a conversão da árvore sintática, criada na etapa anterior, em uma representação intermediária do código fonte.

- Otimização do Código (Opcional)
  - Nesta etapa o código é otimizado para uma determinada arquitetura (hardware e sistema operacional específico).

- Geração de Código Final
  - Nesta última etapa da compilação, o arquivo executável (.exe, por exemplo) é criado, otimizado para aquela arquitetura.



Sintaxe x Semântica

- Sintaxe
  - É a forma como as instruções de uma linguagem são escritas, mas sem atender ao seu significado.
  - Por exemplo: Enquanto na linguagem C, os blocos de comando que serão executados são limitados por “{ }”, em Pascal, eles são limitados pelas palavras “begin” e “end”.
- Semântica
  - É complementar a sintaxe. Ela corresponde à descrição do significado das instruções válidas de uma linguagem. Por exemplo, a sintaxe da instrução if da linguagem C é: if () {} e sua semântica é: “se o valor da expressão for verdadeiro, as instruções incorporadas serão executadas pelo programa”.

Processo de interpretação

- O programa conversor recebe a primeira instrução do programa fonte, confere para ver se está escrita corretamente, converte-a em linguagem de máquina e então ordena ao computador que execute esta instrução.

- Depois repete o processo para a segunda instrução, e assim sucessivamente, até a última instrução do programa fonte.

- Quando a segunda instrução é trabalhada, a primeira é perdida, isto é, apenas uma instrução fica na memória em cada instante.

- Ao final do processo, nenhum código é gerado ou salvo.

Como devemos avaliar um algoritmo?

- 3 critérios para avaliar algoritmos:
  - Corretude;
  - Simplicidade;
  - Eficiência.

Como devemos avaliar um algoritmo?

- Corretude
  - Um algoritmo é correto se para toda entrada especificada a saída correta é produzida;
  - Não basta funcionar para uma determinada entrada específica!
  - Necessidade de testes de exaustão para verificação de corretude.

Como devemos avaliar um algoritmo?

- Simplicidade
  - Um algoritmo é simples se puder ser facilmente entendido, implementado e mantido;
  - Código-fonte claro, devidamente comentado e com indentação (recuo de texto diferenciado em relação à margem esquerda) são essenciais.

Como devemos avaliar um algoritmo?

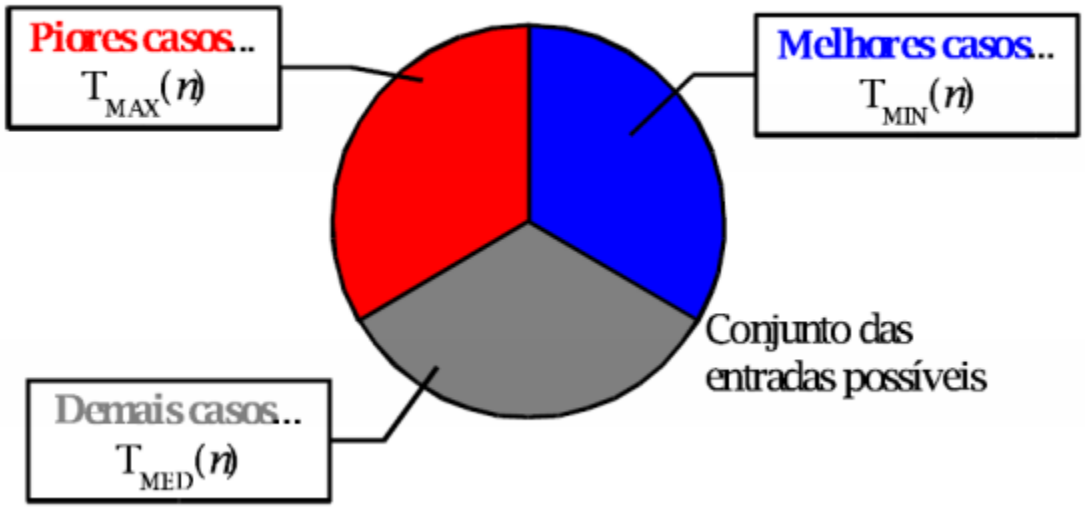
- Eficiência
  - A eficiência de um algoritmo é definida como a inversa da quantidade de recursos computacionais que requer para seu funcionamento;

Como devemos avaliar um algoritmo?

- Vamos nos concentrar na análise de Eficiência!
  - É intuitivo que a eficiência deve admitir uma escala de valores. Portanto, não se trata apenas de verificar se um algoritmo é ou não eficiente... mas, sim, de dizer o quão eficiente ele é.

Eficiência de um algoritmo

- Tipicamente, são identificados 3 casos:
  - Pior caso
  - Melhor caso
  - Demais casos
- Portanto, identificamos uma função para cada caso



Referências

- AdrielCafé.com
  - http://adrielcafe.com/artigos/18-processo-de-compilacao
- Notas de Aula – Disc. Estrutura de Dados – Prof. Claudio Campelo
  - http://www.cin.ufpe.br/~jndm/edados/slides/ClaudioCampelo/AulaIntroducaoAnaliseDeAlgoritmos.pdf

FIM