Disciplina:
Programação Computacional

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

Aula 05: Introdução aos Algoritmos:

- Conceito de algoritmo;
- Conceito de pseudo linguagens;
- Algoritmos estruturados;
- 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.

"O conceito central da programação e da Ciência da Computação é o conceito de algoritmos, isto é, programar é basicamente construir algoritmos."

Niklaus Wirth, *criador da linguagem Pascal*

Algoritmo

- Um **algoritmo** é uma sequência finita de instruções bem definidas e não ambíguas, cada uma das quais pode ser executada num período de tempo finito a fim de alcançar um objetivo

- Um algoritmo não representa, necessariamente, um programa de computador, e sim, os passos necessários para realizar uma tarefa

Algoritmo para Solução de um Problema

Problema do Lobo, o Carneiro e a Alface

- Era uma vez um fazendeiro que foi ao mercado e comprou um lobo, um carneiro, e uma alface. No caminho para casa, o fazendeiro chegou à margem de um rio e arrendou um barco. Mas, na travessia do rio por barco, o agricultor poderia levar apenas a si mesmo e uma única de suas compras - o lobo, o carneiro, ou a alface.

- Se fossem deixados sozinhos em uma mesma margem, o lobo comeria o carneiro, e o carneiro comeria a alface.

- O desafio do fazendeiro é atravessar a si mesmo e as suas compras para a margem oposta do rio, deixando cada compra intacta . Como ele fará isso?

Algoritmo para Solução do Problema do Lobo, o Carneiro e a Alface

Viagem nº	Margem de saída	Viagem	Margem de chegada
(início)	fazendeiro lobo carneiro alface		
1	lobo alface	fazendeiro carneiro →	
2	lobo alface	←fazendeiro	carneiro
3	alface	fazendeiro lobo →	carneiro
4	alface	← fazendeiro carneiro	lobo
5	carneiro	fazendeiro alface →	lobo
6	carneiro	← fazendeiro	lobo alface
7		fazendeiro carneiro →	lobo alface
(fim)			fazendeiro lobo carneiro alface

Ex. de Algoritmo - Receita de Bolo (1)

Adicione os seguintes ingredientes:

- Ovos;
- Farinha de Trigo;
- Leite;
- Açúcar;
- Fermento em pó;

Misture

Leve ao Forno

Pouco detalhado!

Ex. de Algoritmo - Receita de Bolo (2)

Adicione os seguintes ingredientes:

- 4- Ovos;
- 2 copos e meio de farinha de trigo;
- 1 copo de leite;
- 2 copos e meio de açúcar;
- 1 colher de fermento em pó;

Misture

Leve ao Forno por 25 minutos.

Um pouco mais detalhado!

Algoritmo – Dicas

- Preste atenção à ordem lógica da execução das tarefas;
- Lembre-se de que ele deve ter um início e fim;
- Ele deve ser completo;
- Deve ter um alto nível de detalhes;
- Cada tarefa é uma instrução, assim, defina-a bem.

Algoritmos Estruturados

- São aqueles **algoritmos** que buscam resolver **problemas** através do uso de um computador utilizando certas **regras lógicas**.

- São criados com base em uma **linguagem de programação** e podem ser escritos de diversas formas.
  - Exemplos: Pascal, C, C++, C#, Java, Python, Delphi, VB.NET, Fortran, Ada...

- Um algoritmo pode utilizar *variáveis* de determinados **tipos de dados** para armazenar valores.
  - Exemplos: Inteiro, Real, Caracter, String, Booleano...

- Um algoritmo pode ser representado pelo chamado **Português Estruturado** (Portugol), que é uma ferramenta que usa combinações de sequências, seleções e repetições.

Linguagem de programação

- Uma **linguagem de programação** é um método padronizado para comunicar instruções para um computador.

- É um conjunto de regras sintáticas e semânticas usadas para definir um programa de computador

Programação estruturada é uma forma de programação de computadores que defende que todos os programas possíveis podem ser reduzidos a apenas três estruturas: sequencia, decisão e iteração.

Tipos de Dados (em algumas linguagens)

Tipos lógicos	
`boolean`	Representam apenas **1 bit** de informação (0 ou 1). Podem assumir apenas os valores `true` e `false`.
Tipos caractere	
`char`	Representam notação de caracteres de **16 bits** (2 bytes) para formato Unicode UTF-16. Podem assumir caracteres entre `'\u0000'` a `'\uffff'` e valores numéricos entre **0** a **65535**.
Tipos numéricos inteiros	
`byte`	Representam números inteiros de **8 bits** (1 byte). Podem assumir valores entre **-128** a **127**.
`short`	Representam números inteiros de **16 bits** (2 bytes). Podem assumir valores entre **-32.768** até **32.767**.
`Int`	Representam números inteiros de **32 bits** (4 bytes). Podem assumir valores entre **-2.147.483.648** até **2.147.483.647**.
`long`	Representam números inteiros de **64 bits** (8 bytes). Podem assumir valores entre **-9.223.372.036.854.775.808** até **9.223.372.036.854.775.807**.
Tipos numéricos reais	
`float`	Representam números reais de **32 bits** com precisão simples. Podem assumir valores de ponto flutuante com formato definido pela especificação IEEE 754.
`double`	Representam números reais de **64 bits** com precisão dupla. Assim como o float. Podem assumir valores de ponto flutuante com formato definido pela especificação IEEE 754.

Conceito de pseudo linguagens

- **Pseudocódigo** é uma forma genérica de escrever um algoritmo, utilizando uma linguagem simples
  - Nativa a quem o escreve, de forma a ser entendido por qualquer pessoa
- Sem necessidade de conhecer a sintaxe de nenhuma linguagem de programação
- É, como o nome indica, um pseudo-código e, portanto, não pode ser executado num sistema real (computador)
  - Apenas emulado (ou simulado) por algumas ferramentas (interpretadores)
  - Ex: VisuAlg (para Portugol)

Algoritmos Estruturados – Forma Geral

Nome:	Identificador do programa
Variáveis:	Variáveis que são utilizadas no programa
Procedimentos:	procedimentos que podem ser utilizados no programa
Funções:	Funções que podem ser utilizados no programa
Bloco de Ações:	As ações que o programa vai executar.

Algoritmos Estruturados – Exemplo

```
Algoritmo Receita_Bolo
Variáveis
    panela,ovos,copo_farinha,copo_acucar,copo_leite,colher_fermento
Procedimentos
    misture, leve_ao_forno
Funções
    espere
Inicio
    ovos:= 4;
    copo_farinha:=2;
    copo_acucar:=1;
    copo_leite:=1;
    panela:= ovos+copo_farinha+copo_acucar+copo_leite;

    misture
    leve_ao_forno
    espere 25

fim
```

Exemplo em pseudo linguagens

```
programa Habilitacao
var
idade:numerico
inicio
escreva ("informe idade:")
leia(idade)
se idade >= 18 entao
   escreva("pode tirar a carteira")
senao
   escreva("não pode tirar a carteira")
fimse
fimalgoritmo
```

```
VARIAVEIS
S,C,I,A,MD:Real;
INÍCIO
S ← 0;
C ← 0;
PARA I← 1 ATÉ 10 FAÇA
  INÍCIO
    Escreva ('Digite um número');
    LEIA (a);
    SE a >=0 ENTÃO
       INÍCIO
         S ← S+a;
         C ← C+1;
       FIM;
    FIM SE;
  FIM;
FIM PARA;
MD ← S/C;
ESCREVER ('A média é:', MD);
FIM.
```

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

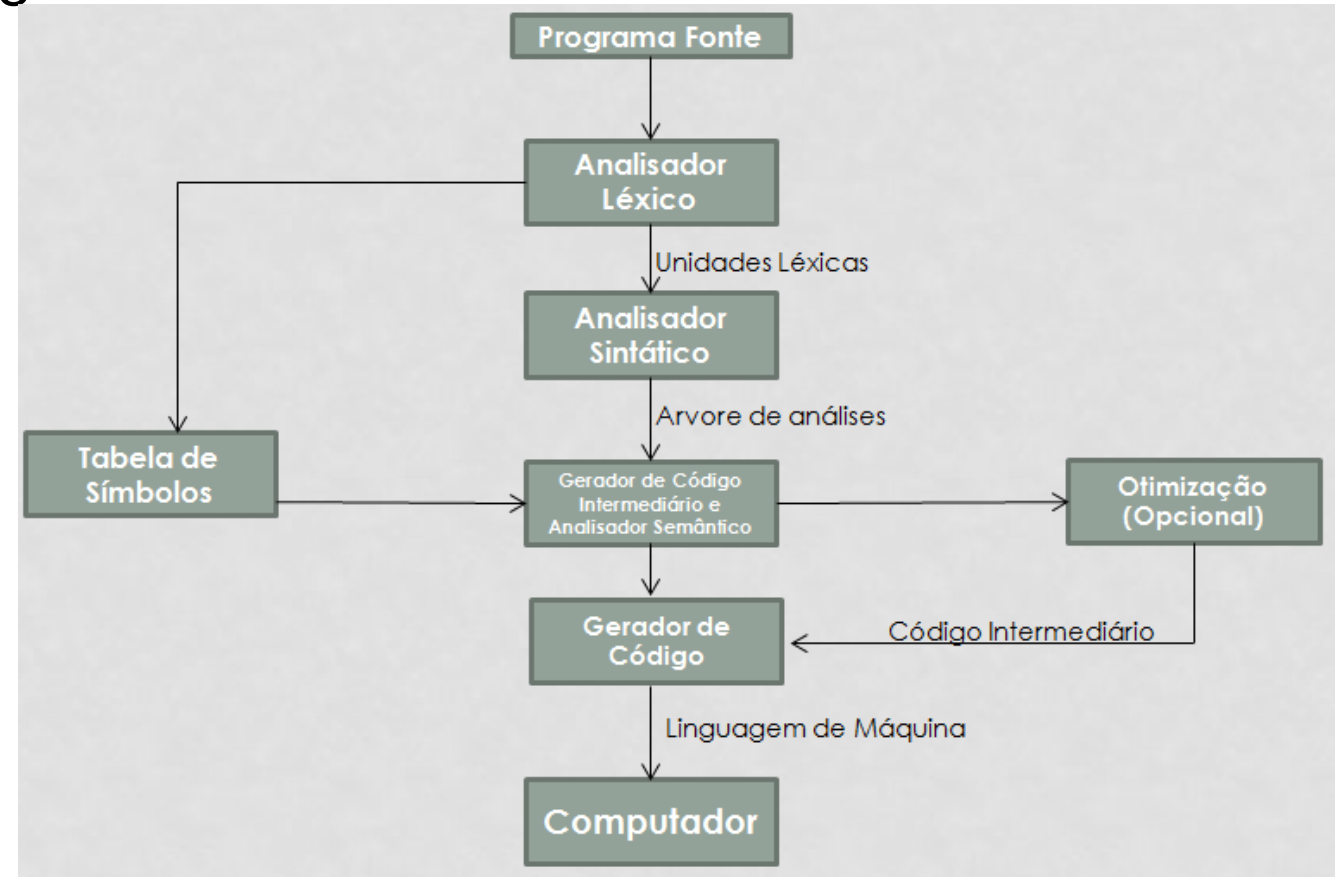
- Compilação é o processo de tradução de um código fonte, escrito normalmente em uma linguagem de alto nível (de fácil entendimento por parte do programador), para uma linguagem de baixo nível (em geral, linguagem de máquina).

- 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

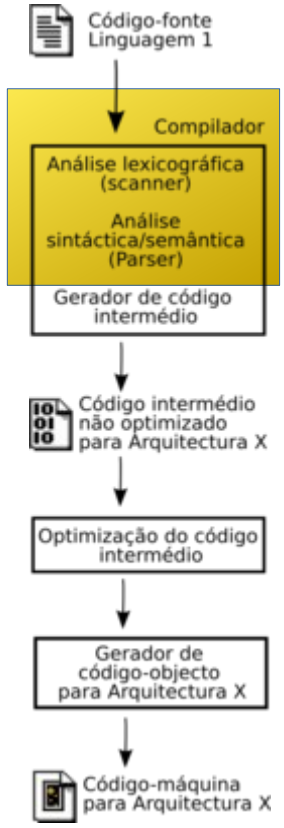


Fases da compilação (I)

- 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.

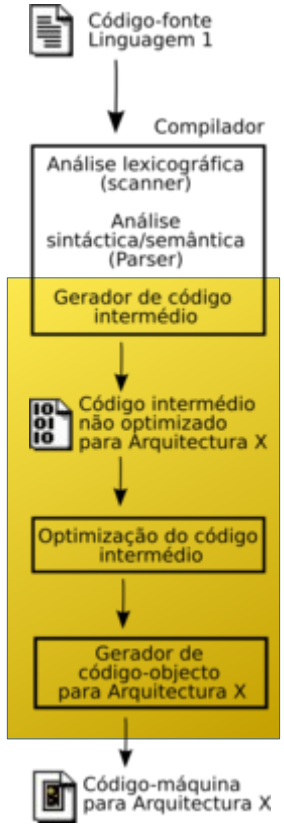


Fases da compilação (II)

- 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.

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