ALGORITMOS: A LÓGICA DA AUTOMATIZAÇÃO DE TAREFAS

Élcio Miguel Pruss[1]

RESUMO

Este artigo apresenta a forma como o algoritmo é estruturado de tal modo a construir uma lógica perfeita em sintaxe e semântica sem alternativas de escapatória o que permite à máquina a execução de tarefas sequenciais automatizadas isentas de erros. Trata-se de elaborar um conjunto de ordens simples com articulação coerente e exata.

Palavras-chave: Algoritmo. Lógica Perfeita. Tarefas Sequenciais Livre de Erros.

ABSTRACT

This article is presented like an algorithm. It is structured in such way in order to build a perfect logic in its syntax and semantic without any other alternatives. It allows the machine to execute automated sequencial chores free of mistakes. It is a matter of elaborating a set of simple orders with coherent and exact articulation.

Key words: Algorithm. Perfect Logic. Sequencial Chores Free of Mistakes.

INTRODUÇÃO

Este artigo trata sobre algoritmos, explicado em duas partes fundamentais: definição e estruturação. Em apoio, intercalam-se exemplos ilustrativos para demostrar sua aplicabilidade e, por último, arremata-se com os resultados desse processo.

A automatização de tarefas é um aspecto marcante da sociedade moderna. O aperfeiçoamento tecnológico alcançado teve, como um dos elementos fundamentais, a análise e a obtenção de descrições da execução de tarefas em termos de ações simples o suficiente, tal que pudessem ser automatizadas por uma máquina especialmente desenvolvida para este fim, o computador (Cormen, 1990).

Para que esta automatização ocorra é necessária uma sequência de instruções que fará com que o computador realize determinada tarefa. Segundo Tremblay & Bunt (1983) esta sequência de instruções não deve possibilitar a interpretação alternativa que possa fazer com que o computador tome um caminho diferente daquele inicialmente planejado.

Este cuidado na formulação das instruções e na sua estruturação é o alvo de estudo dos algoritmos, parte da ciência da computação que desenvolve e aprimora técnicas de construção de programas de forma a determinar que o computador siga pelo único caminho correto possível que conduza aos resultados desejados.

[1] Analista de Tecnologia da Informação. Bacharel em Informática, graduado pela Universidade Positivo. Mestre em Engenharia de Produção pela Universidade Federal de Santa Catarina. Leciona a disciplina de Gestão do Conhecimento e Inteligência Empresarial nas Faculdades Integradas Santa Cruz. C-eletrônico: elcio@santacruz.br.

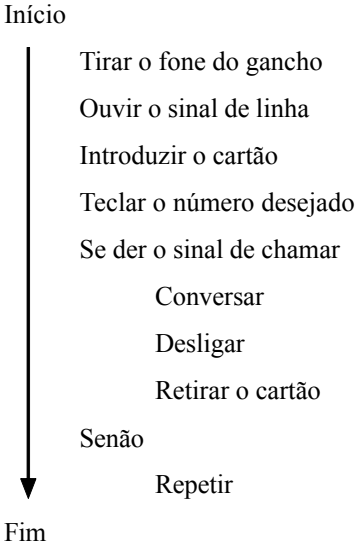
1 DEFINIÇÃO DE ALGORITMO

O algoritmo pode ser usado como uma ferramenta genérica para representar a solução de tarefas independente do desejo de automatizá-las, mas em geral está associado ao processamento eletrônico de dados, onde representa o rascunho para programas ou *softwares*.

Apesar do termo ser novo em si, o conceito é certamente bastante familiar. As indicações dadas para se chegar até uma determinada rua constituem um algoritmo. Uma receita de cozinha é uma forma muito familiar de algoritmo. Uma planta de obra serve ao mesmo propósito num projeto de construção. Estes algoritmos são conhecidos como não computacionais. Um exemplo de um algoritmo não computacional cujo objetivo é usar um telefone público é apresentado

Para que esta automatização ocorra é necessária uma sequência de instruções que fará com que o computador realize determinada tarefa.

a seguir:



Salveti (1998) define um algoritmo como uma sequência finita de instruções ou operações básicas (operações definidas sem ambiguidade e executáveis em tempo finito) cuja finalidade é resolver um problema computacional.

Segundo Pinto (1990) um algoritmo é uma receita para um processo computacional e consiste de uma série de operações primitivas, interconectadas devidamente, sobre um conjunto de objetos. Os objetos manipulados por essas receitas são as variáveis.

Para Terada (1991) algoritmo é uma descrição passo a passo de como um problema é solucionável. Esta descrição deve ser finita, e os passos devem ser bem definidos e sem ambiguidades. Warnier (1991) ressalta que a característica mais importante de um algoritmo é a simplicidade e a isenção de ambiguidade. As instruções devem estar numa ordem cuidadosamente definida e o algoritmo deve ser efetivo, isto é, deve sempre resolver um problema utilizando um número finito de instruções.

Szwarcfiter & Markezon (1994) define algoritmo como um processo sistemático para a resolução de um problema. O desenvolvimento de algoritmos é particularmente importante para problemas a serem solucionados em um computador, pela própria natureza do instrumento utilizado. Dois aspectos devem ser considerados no desenvolvimento de algoritmos: a *correção* e a *análise*.

O primeiro consiste em verificar a exatidão do método empregado, o que é realizado por meio de uma prova matemática. A análise tem por objetivo a obtenção de parâmetros que possam avaliar a eficiência do algoritmo em termos de tempo de execução e memória ocupada. A análise é realizada através de um estudo do comportamento do algoritmo.

Como qualquer modelo, um algoritmo é uma abstração da realidade. Wirth (1989) define abstração como uma simplificação dos fatos constantes da realidade que se encontra o problema a ser resolvido. Os dados utilizados nos algoritmos representam algumas propriedades e características dos fatos reais, pois outras são desprezadas, por serem inexpressivas ou irrelevantes para a solução adotada.

Todas as definições apresentadas de algoritmo nos conduzem a descrição de soluções de problemas do mundo real, para serem desenvolvidas utilizando os recursos computacionais.

Como o mundo computacional possui severas limitações em relação ao real, exige-se que, sejam impostas algumas regras e utilizadas algumas estruturas adequadas para a solução de problemas. Estas estruturas possibilitam a simulação de rotinas, de decisões e de formas de controle.

2 ESTRUTURA BÁSICA DE UM ALGORITMO

Para que o computador possa compreender os passos necessários para a resolução de um determinado problema, devemos estruturar estes passos de forma que sejam perfeitamente compreendidos e executados.

Esta organização das ações a serem tomadas pela máquina de forma organizada e lógica possui uma rígida regra de sintaxe e semântica. Para Pinto (1990) sintaxe é um conjunto de regras formais, que especificam a composição de algoritmos a partir de letras, dígitos e outros símbolos. As regras de semântica especificam o significado de qualquer algoritmo sintaticamente válido, escrito em uma linguagem.

Segundo Salvetti & Barbosa (1998), Pinto (1990), Farrer (1999) e Guimarães (1985) a linguagem mais adequada para representar um algoritmo é a conhecida como pseudocódigo.

Em pseudocódigo a estrutura de um algoritmo é:

```
ALGORITMO <<nome do algoritmo>>
    <<definições das variáveis>>
INÍCIO
    <<comandos 1>>
    <<comandos 2>>
    <<comandos N>>
FIM
```

Esta sintaxe permite definir as variáveis que serão utilizadas na resolução do problema e os comandos que serão aplicados para que possam processar as informações de entrada transformando-as em informações de saída.

2.1 VARIÁVEIS

Variáveis são espaços de memória reservados para armazenar informações. Segundo Sebesta (2000) estes espaços possuem endereço específico, e representam células elementares que contém um valor que representa uma variável. O tipo de informação que será armazenada em uma variável deve ser previamente definido na construção do algoritmo.

Existem vários tipos de dados e os mesmos podem ser definidos de maneira distintos nas diversas linguagens de programação existentes. Segundo Wirth (1989) os tipos de dados mais comuns utilizados em algoritmos são:

TipoExemplo	Inteiro	Real	Caracter	Texto (*String*)	Booleana (*Boolean*)
	2 – 50	2,762 – 100,09	"A" - "4"	"Algoritmo" - "X"	Verdadeiro ou Falso

Tipos de dados

O tipo do dado é associado a um nome que representará a variável. Este nome é formado por uma letra ou então por uma palavra que signifique o conteúdo armazenado. Na visão de Farrer (1999) não se permite o uso de caracteres especiais (acentos), espaços em branco ou de qualquer outro caractere, que não seja letra ou dígito, na formação do nome da variável.

São alguns exemplos válidos de variáveis:

Numero: inteiro

Nota: real

Nome: texto

Situação: *boolean*

3 ESTRUTURAS

As estruturas servem para conduzir o fluxo dos dados nos algoritmos, através de testes e condições. Como destaca Szwarcfiter & Markezon (1994), elas diferem umas das outras pela disposição ou manipulação de seus dados ou variáveis. A disposição dos dados em uma estrutura obedece a condições preestabelecidas e caracteriza a estrutura.

Esta organização das ações a serem tomadas pela máquina de forma organizada e lógica possui uma rígida regra de sintaxe e semântica.

Segundo reforça Sebesta (2000) e Salvetti & Barbosa (1998) estão divididas em estruturas de decisão e de controle. As de decisão testam condições que tomam caminhos específicos sem retorno. Elas optam entre duas ou mais possibilidades. As de controle ou de repetição permitem executar mais de uma vez um determinado número de comandos.

Ainda, Salvetti & Barbosa (1998) afirmam que as estruturas de controle servem para executar uma instrução ou um conjunto de instruções repetidamente dependendo da condição determinada no algoritmo. O processo de repetição é conhecido como laço ou *loop*.

3.1 ESTRUTURA SE-ENTÃO-SENÃO

É uma estrutura de decisão. Inicia com a palavra especial "SE", seguida pela condição a ser testada. A alternativa a ser tomada se a condição for verdadeira é precedida pela palavra especial "ENTÃO". A alternativa a ser tomada se a condição é falsa é precedida pela palavra especial "SENÃO".

Salvetti & Barbosa (1998) e Manber (1989) apresentam esta estrutura na seguinte forma:

SE <<condição>> ENTÃO

<<comandos 1>>

<<comandos N>>

SENÃO

<<comandos 1>>

<comandos N>>

FIMSE

Um exemplo do uso desta estrutura está descrito a seguir:

Se João for maior de 18 anos deve ir votar, se ele não for pode ficar em casa. A condição: **João maior que 18 anos** determina uma ação para João, a de ir votar. A representação em uma estrutura de condição é:

SE idade de João > 18 ENTÃO

João deve votar

```
SENÃO
        João pode ficar em casa
FIMSE
```

3.2 ESTRUTURA ESCOLHA-CASO

É uma estrutura de decisão. É utilizada quando existem diversas opções a serem seguidas dependendo do que o usuário ou o programa solicite. Segundo Ziviani (1999) e Salvetti & Barbosa (1998) esta estrutura é apresentada na seguinte forma:

```
CASO <<condição>> FAÇA
        opção 1: <<comandos 1>>
        opção N: <<comandos N>>
SENÃO
        <<comandos 1>>
        <<comandos N>>
FIMCASO
```

O exemplo a seguir facilita a compreensão desta estrutura:

```
CASO opção ENTÃO
        1: Veja o Saldo
        2: Veja o Extrato
        3: Permita o Depósito
        4: Sair
SENÃO
        Opção Inválida
FIMCASO
```

Lê-se da seguinte maneira: caso a opção seja 1 execute o saldo, caso seja 2 execute o extrato, caso seja 3 execute o depósito, caso seja 4 saia e se a opção não for nenhuma das opções válidas a escolha será inválida. A estrutura ESCOLHA-CASO também pode ser representada pela estrutura SE-ENTÃO-SENÃO alinhadas, mas é inviável utilizar, pois o código ficaria de difícil leitura:

```
SE opção = 1 ENTÃO
        Saldo
SENÃO
        SE opção = 2 ENTÃO
                Extrato
        SENÃO
                SE opção = 3 ENTÃO
                        Deposito
                SENÃO
```

```
                    Sair
                FIMSE
            FIMSE
        FIMSE
```

O exemplo acima evidencia o aumento da complexidade da leitura se fosse utilizada a estrutura SE-ENTÃO-SENÃO.

3.3 ESTRUTURA ENQUANTO-FAÇA

Esta estrutura de repetição permite que enquanto uma determinada condição for verdadeira ou válida os comandos pertencentes à estrutura sejam executados. O término da execução dos comandos está vinculado à condição se tornar falsa. A sintaxe a seguir representa a estrutura ENQUANTO-FAÇA, definida por Sebesta (2000) e Terada & Setzer (1992):

```
ENQUANTO <<condição>> FAÇA
    <<comandos 1>>
    <<comandos 2>>
    <<comandos N>>
FIMENQUANTO
```

O exemplo abaixo utiliza a estrutura ENQUANTO-FAÇA e escreve na tela do computador os números inteiros de 1 até 200.

```
contador := 1
ENQUANTO contador <= 200 FAÇA
    escreva (contador)
    contador := contador + 1
FIMENQUANTO
```

A variável definida como contador será inicializada com o valor 1 e acrescida de uma unidade toda vez que escrever o valor na tela até chegar no valor 200, quando então o laço será finalizado pois a condição estabelecida se tornará falsa. O nome dado a variável que controla a execução do laço é variável de controle (Terada & Setzer, 1992; Guimarães, 1985).

3.4 ESTRUTURA PARA-FAÇA

Segundo Ziviani (1999) é uma estrutura de controle. É uma simplificação da estrutura ENQUANTO-FAÇA, pois não necessita do incremento da variável de controle, que será automaticamente incrementada na definição da estrutura. A sintaxe abaixo representa a estrutura PARA-FAÇA:

```
PARA <<variável>>:=<<valor inicial>> ATE <<valor final>> FAÇA
    <<comandos 1>>
    <<comandos N>>
FIMPARA
```

O incremento definido na estrutura acima é por padrão uma unidade. O exemplo utilizado no item anterior ficaria da seguinte forma utilizando a estrutura FAÇA-PARA:

```
PARA contador := 1 ATÉ 200  FAÇA
        escreva (contador)
FIMPARA
```

Nesta estrutura, conforme afirma Guimarães (1985), a variável contador é inicializada com valor 1 e é acrescida de uma unidade até o valor 200. A forma de inicialização e a taxa de acréscimo já estão definidas na própria estrutura e não necessitam de atribuições em linhas de comandos individuais. Esta estrutura em comparação a ENQUANTO-FAÇA é mais simplificada, porém executa o laço do valor inicial até o final sem interrupção. Caso haja a necessidade de, por algum motivo, o laço seja interrompido a estrutura mais adequada é a ENQUANTO-FAÇA.

3.5 ESTRUTURA REPITA-ATÉ

É uma estrutura de controle. Assim como a estrutura ENQUANTO-FAÇA que é usada para repetir diversas vezes uma ou mais instruções, a estrutura REPITA-ATÉ também pode ser aplicada para esta finalidade. A diferença está na validação ou condição, que nesta estrutura, REPITA-ATÉ, é no final, fazendo com que os comandos internos a estrutura sejam executados, pelo menos, uma vez. Sebesta (2000), Salvetti & Barbosa (1998) e Terada & Setzer (1992) apresentam esta estrutura na seguinte forma:

```
REPITA
        <<comandos 1>>
        <<comandos 2>>
        <<comandos N>>
ATE <<condição>>
```

Possui a mesma característica da estrutura ENQUANTO-FAÇA que pode ter a execução do laço interrompida. O exemplo utilizado nos itens anteriores ficaria da seguinte forma utilizando a estrutura REPITA-ATÉ:

```
contador := 1
REPITA
        escreva (contador)
        contador := contador + 1
ATÉ contador = 201
```

A variável definida como contador recebe o valor inicial 1. Este valor é escrito na tela do computador pelo comando escreva e em seguida seu valor é acrescido de uma unidade, quando então será feito o teste da condição para a execução do laço. Neste caso se a condição fosse falsa o comando de escrever na tela e o de acrescer a variável de controle já teria sido executado, pelo menos, uma vez.

CONCLUSÃO

A programação está apoiada sobre estruturas de decisão e controle. A combinação e o alinhamento entre elas formam a estrutura de um programa ou *software*, que é transformação do algoritmo em uma linguagem comercial, industrial ou científica.

A habilidade de abstrair um problema e encontrar a estrutura de programação adequada para resolvê-lo é a habilidade desenvolvida nas disciplinas de algoritmos nos cursos da área de tecnologia.

REFERÊNCIAS

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L. **Introduction to algorithms**. EUA: McGraw-Hill Book Company, 1990.

FARRER, Harry *et al.* **Pascal estruturado**. 3. ed. Rio de Janeiro: LTC, 1999.

GUIMARÃES, Ângelo de Moura; LAGES, Newton Alberto de Castilho. **Algoritmos e estruturas de dados**. Rio de Janeiro: LTC, 1985.

MANBER, Udi. **Introduction to algorithms**: a creative approach. EUA: Addison-Wesley Company Inc., 1989.

PINTO, Wilson Silva. **Introdução ao desenvolvimento de algoritmos e estrutura de dados**. São Paulo: Érica, 1990.

SALVETTI, Dirceu Douglas; BARBOSA, Lisbete Madsen. **Algoritmos**. São Paulo: Makron Books, 1998.

SEBESTA, Robert W. **Conceito de linguagens de programação**. Tradução de José Carlos Barbosa dos Santos. 4. ed. Porto Alegre: Bookman, 2000.

SZWARCFITER, Jayme Luiz; MARKEZON, Lilian. **Estruturas de dados e seus algoritmos**. 2. ed. Revista. Rio de Janeiro: LTC, 1994.

TERADA, Routo. **Desenvolvimento de algoritmos e estruturas de dados**. São Paulo: McGraw-Hill, Makron, 1991.

_____; SETZER, Valdemar W. **Introdução à computação e à construção de algoritmos.** São Paulo: Makron Books, 1992.

TREMBLAY, Jean-Paul; BUNT, Richard B. **Ciência dos computadores:** uma abordagem algorítmica. São Paulo: McGraw-Hill do Brasil, 1983.

WARNIER, Jean Dominique. **LCP**: lógica de construção de programas. Rio de Janeiro: Campus, 1991.

WIRTH, Niklaus. **Algoritmos e estrutura de dados**. Rio de Janeiro: Prentice-Hall do Brasil, 1989.

ZIVIANI, Nivio. **Projeto de algoritmos:** com implementações em Pascal e C. 4. ed. São Paulo: Pioneira, 1999.