1. ALGORITMOS

1.1. DEFINIÇÃO DE ALGORITMO

A palavra ***algoritmo***, à primeira vista, parece-nos estranha. Embora possua designação desconhecida, fazemos uso constantemente de algoritmos em nosso cotidiano: a maneira como uma pessoa toma banho é um algoritmo. Outros algoritmos frequentemente encontrados são:

- instruções para se utilizar um aparelho eletrodoméstico;
- uma receita para preparo de algum prato;
- guia de preenchimento para declaração do imposto de renda;
- a regra para determinação de máximos e mínimos de funções por derivadas sucessivas;
- a maneira como as contas de água, luz e telefone são calculadas mensalmente; etc.

São vários os conceitos para algoritmo. Escolhemos alguns para serem apresentados aqui:

> *"Um conjunto finito de regras que provê uma sequência de operações para resolver um tipo de problema específico"*
> [KNUTH]

> *"Sequência ordenada, e não ambígua, de passos que levam à solução de um dado problema"*
> [TREMBLAY]

> *"Processo de cálculo, ou de resolução de um grupo de problemas semelhantes, em que se estipulam, com generalidade e sem restrições, as regras formais para a obtenção do resultado ou da solução do problema"* [AURÉLIO]

1.2. POR QUE PRECISAMOS DE ALGORITMOS?

Vejamos o que algumas pessoas importantes, para a Ciência da Computação, disseram a respeito de algoritmo:

> *"A noção de algoritmo é básica para toda a programação de computadores".*
> [KNUTH - Professor da Universidade de Stanford, autor da coleção "The art of computer programming"]

> *"O conceito central da programação e da ciência da computação é o conceito de algoritmo".*
> [WIRTH - Professor da Universidade de Zurique, autor de diversos livros na área e responsável pela criação de linguagens de programação como ALGOL, PASCAL e MODULA-2]

A importância do algoritmo está no fato de termos que especificar uma seqüência de passos lógicos para que o computador possa executar uma tarefa qualquer, pois o mesmo por si só não tem vontade própria, faz apenas o que mandamos. Com uma ferramenta algorítmica, podemos conceber uma solução para um dado problema, independendo de uma linguagem específica e até mesmo do próprio computador.

1.3. CARACTERÍSTICAS

Todo algoritmo deve apresentar algumas características básicas:

- Ter fim;
- Não dar margem à dupla interpretação (não ambíguo);
- Capacidade de receber dado(s) de entrada do mundo exterior;
- Poder gerar informações de saída para o mundo externo ao do ambiente do algoritmo;

- Ser efetivo (todas as etapas especificadas no algoritmo devem ser alcançáveis em um tempo finito).

1.4. FORMAS DE REPRESENTAÇÃO

Algoritmos podem ser representados, dentre outras maneiras, por:

1.4.1. DESCRIÇÃO NARRATIVA

Faz-se uso do português para descrever algoritmos.

EXEMPLO: Receita de Bolo:

Providencie manteiga, ovos, 2 Kg de massa, etc.
Misture os ingredientes
Despeje a mistura na fôrma de bolo
Leve a fôrma ao forno
Espere 20 minutos
Retire a fôrma do forno
Deixe esfriar
Prove

VANTAGENS:

- O português é bastante conhecido por nós; DESVANTAGENS:
- Imprecisão;
- Pouca confiabilidade (a imprecisão acarreta a desconfiança);
- Extensão (normalmente, escreve-se muito para dizer pouca coisa).

1.4.2. FLUXOGRAMA

Utilização de símbolos gráficos para representar algoritmos. No fluxograma existem símbolos padronizados para início, entrada de dados, cálculos, saída de dados, fim, etc.

Cálculo Decisão Entrada Saída Início/Fim

EXEMPLO	EXPLICAÇÃO
Início	Início do algoritmo
NUM	Entrada do número
DOBRO = NUM * 2	Cálculo do dobro do número
DOBRO	Apresentação do resultado
Fim	Fim do algoritmo

VANTAGENS:

- Uma das ferramentas mais conhecidas;
- Figuras dizem muito mais que palavras;
- Padrão mundial DESVANTAGENS:

- Pouca atenção aos dados, não oferecendo recursos para descrevê-los ou representá-los; • Complica-se à medida que o algoritmo cresce.

1.4.3. LINGUAGEM ALGORÍTMICA

Consiste na definição de uma pseudolinguagem de programação, cujos comandos são em português, para representar algoritmos.

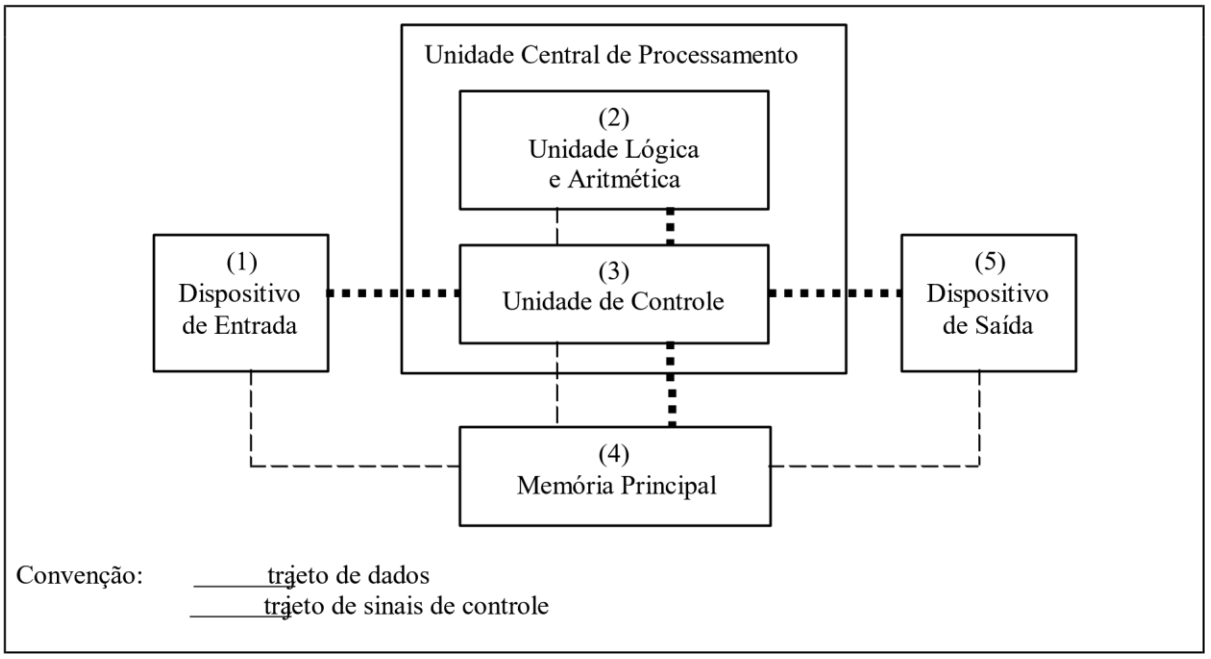
EXEMPLO:
```
Algoritmo CALCULA_DOBRO
  NUM,DOBRO : inteiro início
Leia NUM
  DOBRO  2 * NUM
Escreva DOBRO fim
```

VANTAGENS:

- Usa o português como base;
- Pode-se definir quais e como os dados vão estar estruturados;
- Passagem quase imediata do algoritmo para uma linguagem de programação qualquer. DESVANTAGENS:
- Exige a definição de uma linguagem não real para trabalho;
- Não padronizado.

1.5. UM AMBIENTE PARA ESCREVER ALGORITMOS

Descreveremos uma máquina hipotética para a qual escreveremos nossos algoritmos. O nosso computador hipotético apresentará a seguinte organização:



Cada uma das partes constituintes da figura acima tem os seguintes significados:

(1) Dispositivo de entrada (o teclado):
É o meio pelo qual os dados que serão trabalhados pelo algoritmo vão ser introduzidos em nosso computador hipotético;

(2) Unidade Lógica e Aritmética (ULA):
Parte responsável pelas operações matemáticas e avaliações lógicas;

(3) Unidade de Controle:
Exerce controle sobre as demais partes do nosso computador. É uma verdadeira gerente que distribui tarefas às outras unidades;

(4) Memória:

Guarda o algoritmo a ser executado e os dados a serem utilizados pelo mesmo. Todo dado fornecido ao computador e o resultado de suas operações ficam guardados na memória;

(5) Dispositivo de Saída (vídeo e impressora):
É o meio que se dispõe para apresentação dos resultados obtidos.

1.5.1. FUNCIONAMENTO DO NOSSO COMPUTADOR

Todos os computadores, independentemente dos seus tamanhos, são conceitualmente semelhantes ao esquema da figura anterior (há algumas diferenças, mas não trataremos aqui dos casos especiais).

Resumidamente, podemos afirmar que existem 4 (quatro) operações básicas que qualquer computador pode executar:

a) **Operações de entrada e saída:** ler dados do teclado e escrever dados na tela são exemplos destas operações. Elas servem para introduzir dados na memória do nosso computador e exibir dados que já estejam lá armazenados;

b) **Operações aritméticas:** são utilizadas na realização de operações matemáticas (adição, subtração, multiplicação e divisão);

c) **Operações lógicas e relacionais:** têm aplicabilidade em comparações, testes de condições lógicas (2>6 ? X=Y ?);

d) **Movimentação de dados entre os vários componentes:** as operações aritméticas são executadas na Unidade Lógica e Aritmética, necessitando da transferência dos dados para essa unidade e da volta do resultado final para ser guardado na memória.

1.5.2. RESOLVENDO UM PROBLEMA

Suponha que queiramos resolver o seguinte problema: a partir de dois números que serão informados, calcular a adição dos mesmos. Se você fosse encarregado de efetuar essa tarefa, seria bem provável que utilizasse os passos a seguir:

a) Saber quais são os números;

b) Calcular a soma dos números;

c) Responder à questão com o valor do resultado.

Vejamos como seria resolvido esse mesmo problema em termos das operações básicas citadas anteriormente:

a) operação de entrada de dados dos números ;

b1) movimento do valor dos números entre a memória e a ULA;

b2) operação aritmética de somar os 2 números;

b3) movimentação do resultado da ULA para guardar na memória;

c) operação de saída do resultado, que está guardado na memória, para o dispositivo de saída desejado.

Deve-se salientar que os passos b1 e b3, normalmente, ficam embutidos na operação matemática, não sendo explicitados.

Em resumo, pode-se dizer que escrever algoritmos ou, em última análise, programar consiste em dividir qualquer problema em muitos pequenos **passos**, usando uma ou mais das quatro operações básicas citadas.

Esses passos que compõem o algoritmo são denominados de **comandos**. Os comandos de uma linguagem de programação podem estar mais próximos da máquina (linguagens de baixo nível) ou serem mais facilmente entendidos pelo homem (linguagens de alto nível). A seqüência de operações básicas, dada anteriormente,

para resolver o problema de adicionar dois números, está em uma linguagem de baixo nível para o nosso computador hipotético. Em uma linguagem de alto nível teríamos um resultado assim:

Leia X,Y

SOMA X + Y

Escreva SOMA

Variáveis

As variáveis representam a memória do computador onde se podem guardar dados de entrada e resultados.
Facilitam a escrita dos programas ao permitirem identificar a memória através de nomes escolhidos pelo utilizador.
Nome da variável: letras, números e

1.6. ESTRUTURAS CHAVES DA CONSTRUÇÃO DE ALGORITMOS

Existem 3 estruturas básicas de controle nas quais se baseiam os algoritmos: sequenciação, decisão e repetição. Detalharemos cada uma delas:

1.6.1. SEQUENCIAÇÃO

Os comandos do algoritmo fazem parte de uma seqüência, onde é relevante a ordem na qual se encontram os mesmos, pois serão executados um de cada vez, estritamente, de acordo com essa ordem. De uma forma genérica, poderíamos expressar uma seqüência da seguinte maneira:

Comando-1
Comando-2
Comando-3
Comando-n

Tem-se uma sequenciação de **n** comandos na qual os comandos serão executados na ordem em que aparecem, isto é, o comando de ordem i+1 só será executado após a execução do de ordem i (o 3º só será executado após o 2º).

Todo algoritmo é uma sequência. A sequenciação é aplicada quando a solução do problema pode ser decomposta em passos individuais.

1.6.2. DECISÃO OU SELEÇÃO

Essa estrutura também é conhecida por estrutura condicional. Há a subordinação da execução de um ou mais comandos à veracidade de uma condição. Vejamos o funcionamento:

Se <condição> então <seq.
de comandos-1> senão
<seq. de comandos-2>

Se a <condição> for verdadeira será executado a <seq. de comandos-1> e, em caso contrário, teremos a execução da
<seq. de comandos-2>.

A decisão deve ser sempre usada quando há a necessidade de testar alguma condição e em função da mesma tomar uma atitude. Em nosso dia-a-dia, estamos sempre tomando decisões, vejamos um exemplo:

Se tiver dinheiro suficiente, **então** vou almoçar em um bom restaurante.
Caso contrário (**senão**), vou comer um sanduíche na lanchonete da esquina.

1.6.3. REPETIÇÃO OU ITERAÇÃO

Essa estrutura também é conhecida por “looping” ou laço. A repetição permite que tarefas individuais sejam repetidas um número determinado de vezes ou tantas vezes quantas uma condição lógica permita. Vejamos alguns exemplos:

a) vou atirar pedras na vidraça até quebrá-la;
b) baterei cinco pênaltis;
c) enquanto tiver saúde e dinheiro, vou desfrutar a vida.

No exemplo (a), vai-se repetir a ação de atirar pedras na janela até que seja satisfeita a condição de quebrar a janela.

No exemplo (b), haverá a repetição da atitude de bater um pênalti um número determinado de vezes (cinco).

No exemplo (c), a condição que me permitirá continuar desfrutando a vida é ter dinheiro e saúde.

A utilização combinada dessas 3 estruturas descritas vai permitir expressar, usando qualquer que seja a ferramenta, a solução para uma gama muito grande de problemas. Todas as linguagens de programação oferecem representantes dessas estruturas.

2. CONCEITOS BÁSICOS DE PROGRAMAÇÃO

Para armazenar um algoritmo na memória de um computador e para que ele possa, em seguida, comandar as operações a serem executadas, é necessário que ele seja **programado**, isto é, que seja transcrito para uma **linguagem** que o computador possa entender, direta ou indiretamente.

2.1. LINGUAGENS DE PROGRAMAÇÃO

Linguagem é uma maneira de comunicação que segue uma forma e uma estrutura com significado interpretável. Portanto, **linguagem de programação** é um conjunto finito de palavras, comandos e instruções, escritos com o objetivo de orientar a realização de uma tarefa pelo computador.

Logicamente, a linguagem que nós utilizamos em nosso cotidiano é diferente da linguagem utilizada pela máquina. A máquina trabalha somente com códigos numéricos (linguagem de máquina), baseados nos números 0 e 1 (sistema binário), que representam impulsos elétricos, ausente e presente.

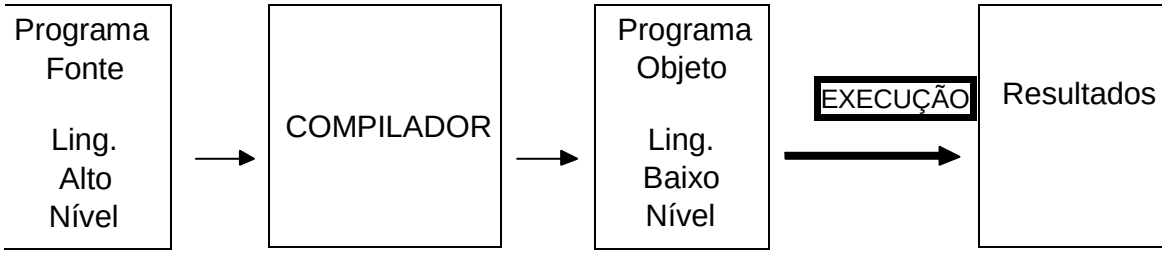
Assim, qualquer linguagem de programação deve estar situada entre dois extremos: o da linguagem natural do homem
(muito clara, porém lenta) e o da linguagem de máquina (muito rápida, porém complexa).

Este é o conceito de nível de linguagem: alto nível para as mais próximas da linguagem humana; baixo nível para as mais semelhantes à linguagem de máquina.

2.2. TRADUTORES

Para que um computador possa "entender" um programa escrito em uma linguagem de alto nível, torna-se necessário um meio de tradução entre a linguagem utilizada no programa e a linguagem de máquina. Este meio pode ser de dois tipos: ***compilador*** e ***interpretador***.

COMPILADOR - traduz o programa escrito em linguagem de alto nível (programa-fonte) para um programa equivalente escrito em linguagem de máquina (programa-objeto).



Exemplos de linguagens de programação: PASCAL, C, CLIPPER, COBOL, HTML, JAVA, etc.