CONCEITO DE ALGORITMO

De forma geral podemos definir o **algoritmo** como uma **sequência finita de ações executáveis para a obtenção de um objetivo bem definido.**

O termo **algoritmo** se originou, provavelmente, do sobrenome do matemático persa Al-Khwarizmi, do século IX, cujas obras foram traduzidas no ocidente no século XII, tendo uma delas recebido o nome ***Algorithmi de numero indorum***, sobre os algoritmos usando o sistema de numeração decimal (indiano).

Os algoritmos fazem parte do dia a dia das pessoas:

- instruções para o uso de medicamentos;
- indicações de como montar um aparelho;
- uma receita de culinária.

A ***execução*** de um algoritmo consiste em repetir determinados passos (fazer iterações) e/ou tomar as decisões (tais como comparações ou operações lógicas) até que a tarefa seja completada.

Um algoritmo não resolverá um problema se estiver implementado incorretamente ou se não for apropriado ao problema.

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

O algoritmo corresponde a uma descrição de um padrão de comportamento, expresso em termos de um conjunto finito de ações (uma sequência de passos).

Sua implementação pode ser feita por um computador, por outro tipo de autômato ou mesmo por um ser humano.

Diferentes algoritmos podem realizar a mesma tarefa usando um conjunto diferenciado de instruções em mais ou menos tempo, espaço ou esforço do que outros.
Essa diferença é reflexo da complexidade computacional aplicada, que depende de estruturas de dados e operações usadas na implementação do algoritmo.

Características fundamentais de um algoritmo

- Um algoritmo deve terminar após um número finito de passos.
- Cada passo de um algoritmo deve ser precisamente definido.
- Deve ser prevista a entrada e saída de dados.
- Um algoritmo deve ser eficiente.

Premissas básicas de um algoritmo eficiente:

- Definir ações simples e sem ambiguidade;
- Organizar as ações de forma ordenada;
- Estabelecer as ações dentro de uma sequência finita de passos.

Tarefas que podem ser algoritmizadas:

- Ler e escrever dados;
- Avaliar expressões algébricas, relacionais e lógicas;
- Tomar decisões com base em resultados das expressões avaliadas;
- Repetir um conjunto de ações de acordo com uma condição.

Método para a construção de algoritmos:

- Compreender completamente o problema a ser resolvido, destacando os pontos mais importantes e os objetos que o compõem;
- Definir os dados de entrada, ou seja, quais dados serão fornecidos e quais objetos fazem parte desse cenário problema;
- Definir o processamento, ou seja, quais cálculos serão efetuados e quais as restrições para esses cálculos. O processamento é responsável pela transformação dos dados de entrada em dados de saída;
- Definir os dados de saída, ou seja, quais dados serão gerados depois do processamento;
- Construir o algoritmo utilizando um determinado tipo de codificação (ex. pseudocódigo);
- Testar o algoritmo realizando simulações.

Exemplos de Algoritmos

Troca de pneu de carro

1: desligar o carro
2: pegar as ferramentas (chave e macaco)
3: pegar o estepe
4: suspender o carro com o macaco
5: desenroscar os parafusos do pneu furado
6: colocar o estepe
7: enroscar os parafusos
8: baixar o carro com o macaco
9: guardar as ferramentas

Troca de lâmpada

1. *pegar uma escada;*
2. *posicionar a escada embaixo da lâmpada;*
3. *subir na escada;*
4. *retirar a lâmpada velha;*
5. *colocar a lâmpada nova.*

Questionamento: “Posso realizar essas atividades de maneira diferente?”

Às vezes um problema pode ser resolvido de diversas maneiras, porém gerando a mesma resposta;

Podem existir vários algoritmos para solucionar o mesmo problema.

Troca de lâmpada - Versão 2

1. *pegar uma escada;*
2. *posicionar a escada embaixo da lâmpada;*
3. *buscar uma lâmpada nova;*
4. *acionar o interruptor;*
5. *se a lâmpada não acender então*
    1. *subir na escada;*
    2. *retirar a lâmpada velha;*
    3. *colocar a lâmpada nova.*

Teste

Troca de lâmpada - Versão 3

1. *acionar o interruptor;*
2. *Se a lâmpada não acender, então*
    1. *pegar uma escada;*
    2. *posicionar a escada embaixo da lâmpada;*
    3. *subir na escada;*
    4. *retirar a lâmpada velha;*
    5. *colocar a lâmpada nova.*

Teste

Exemplo

Digamos que existam duas vasilhas com capacidades de 9 e 4 litros, respectivamente.

Essas vasilhas não possuem nenhum tipo de marcação, de modo que não é possível ter medidas como metade ou um terço.

Considerando isso, apresente uma sequência de passos, em que, usando as vasilhas de 9 e 4 litros seja possível encher uma terceira vasilha de medida desconhecida com **6** litros de água.

Solução:

1. Encha a vasilha de 9 litros;
2. Usando a vasilha de 9 litros, encha a vasilha de 4 litros;
3. Despeje o que sobrou na vasilha de 9 litros (5 litros) na terceira vasilha. Observe que falta um litro para completar os seis litros;
4. Esvazie a vasilha de 4 litros;
5. Encha a vasilha de 9 litros;
6. Usando a vasilha de 9 litros encha a vasilha de 4 litros;
7. Esvazie a de 4 litros;
8. Usando o que restou na vasilha de 9 litros (5 litros), encha novamente a vasilha de quatro litros;
9. Despeje o que sobrou na vasilha de 9 litros (1 litro) na terceira vasilha, que agora tem **6** litros.

TIPOS DE REPRESENTAÇÃO DE ALGORITMOS

- Descrição narrativa
- Fluxograma
- Diagrama de Chapin
- Pseudocódigo ou Portugol

Descrição Narrativa

Consiste em analisar o enunciado do problema e escrever, utilizando uma linguagem natural (ex. a língua portuguesa), os passos a serem seguidos para sua resolução

Vantagem: não é necessário aprender nenhum conceito novo

Desvantagem: a linguagem natural abre espaço para várias interpretações, o que posteriormente dificultará a codificação desse algoritmo

Fluxograma

Consiste em analisar o enunciado do problema e escrever, utilizando símbolos gráficos predefinidos, os passos a serem seguidos para sua resolução

Vantagem: o entendimento de elementos gráficos é mais simples que o entendimento de textos

Desvantagem: é necessário aprender a simbologia dos fluxogramas e, além disso, o algoritmo resultante não apresenta muitos detalhes, dificultando sua codificação.

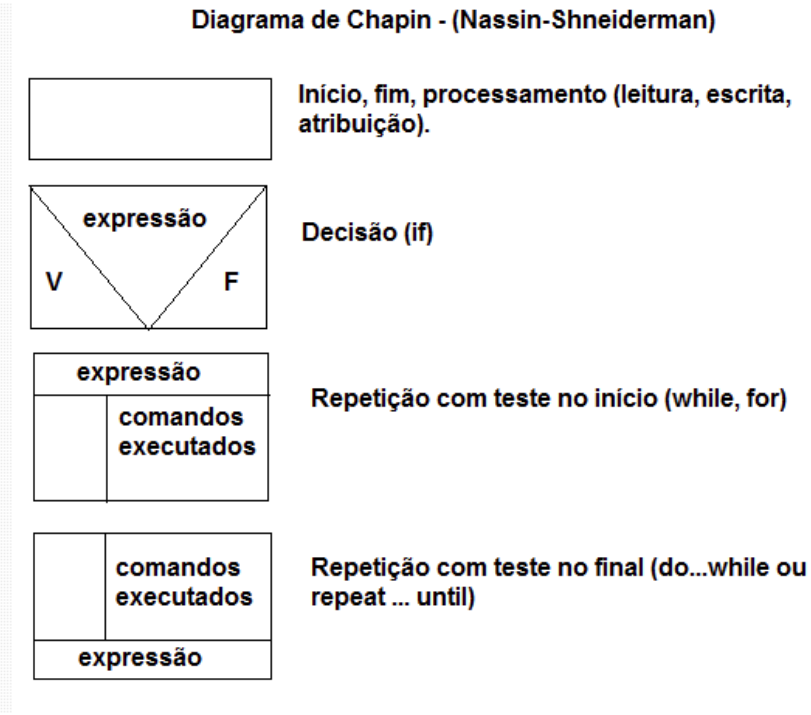
Diagrama de Chapin

O Diagrama de Chapin também conhecido como Diagrama de Nassi-Shneiderman tal como o fluxograma, permite a visualização do fluxo lógico do algoritmo e é voltado para a programação estruturada.

Vantagem: o entendimento de elementos gráficos é mais simples que o entendimento de textos.

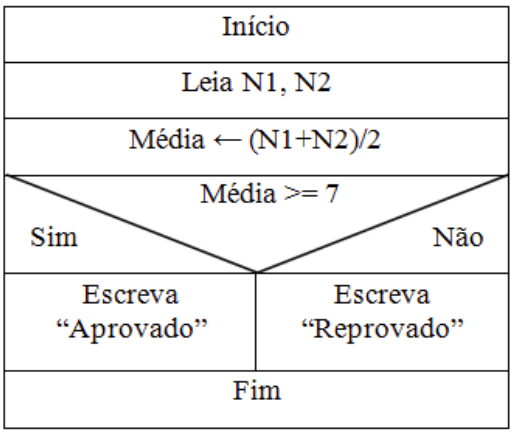
Desvantagem: é necessário aprender uma simbologia específica.

Simbologia utilizada:



Exemplo:

Exemplo do Diagrama de Chapin para calcular a média de dois números e dependendo do resultado imprime uma mensagem "*Aprovado*" ou "*Reprovado*".



Pseudocódigo ou Portugol

Consiste em analisar o enunciado do problema e escrever, por meio de regras, predefinidas, os passos a serem seguidos para sua resolução

Vantagem: a passagem do algoritmo para qualquer linguagem de programação é quase imediata, bastando conhecer as palavras reservadas da linguagem a ser utilizada

Desvantagem: é necessário aprender as regras do pseudocódigo

Exemplo de Algoritmo em Pseudocódigo

Algoritmo de multiplicação de dois números:

```
INICIO_ALGORITMO
DECLARE N1, N2, M  NUMÉRICO
ESCREVA “Digite dois números”
LEIA N1, N2
M ← N1 * N2
ESCREVA “Multiplicação = ”, M
FIM_ALGORITMO
```

Exercícios:

1. Imagine que uma pessoa decida ir de táxi a uma reunião de negócios.

Monte um algoritmo com a sequência de ações para que ela chegue ao prédio onde vai ocorrer a reunião.

a) Entrar no prédio da reunião.

b) Sair do táxi.

c) Acenar para que o táxi pare.

d) Perguntar o preço da corrida.

e) Informar o destino ao motorista.

f) Esperar o táxi.

g) Pagar a corrida.

h) Entrar no táxi.

2. Monte um algoritmo com as ações para retirar R$ 100,00 de um caixa automático de banco.

Programação estruturada

Um dos principais objetivos no desenvolvimento de software é construir programas rápidos, corretos e econômicos.

Esse objetivo guia a criação e a evolução dos paradigmas de desenvolvimento de software até o presente momento.

A programação estruturada (ou modular) sugeri que todos os programas possíveis podem ser escritos utilizando apenas três recursos:

- sequência
- decisão
- iteração (repetição)

Na realidade esses mecanismos estão sintonizados com as instruções do CPU (e com as operações da máquina de Turing).

Portanto o processador sempre executa "um programa estrutural" mesmo que as instruções que ele ler da memória não fazem parte de um programa estrutural.

Em programação estruturada os verbos em uma especificação de sistema ajudam o programador a determinar o conjunto de funções que trabalharão juntas para implementar o sistema.

No processo de desenvolvimento de software o objetivo é visualizar as tarefas a serem executadas.

Os grupos de ações que executam alguma tarefa então são reunidos em funções e funções são agrupadas para formar programas.

Os programadores concentram-se em escrever funções. Os dados são certamente importantes, mas a visão é que os dados existem principalmente por causa das ações que as funções executam.

De ponto de vista de reutilização de software – a parte reaproveitada são as funções.

Introdução a Técnicas Computacionais

Cada programador possui suas próprias características para escrever um programa de computador.

Normalmente é desejável que um programa possa ser compreendido por outras pessoas:

- Escrito de forma clara
- Seguindo um determinado padrão de programação, independente da linguagem usada

Etapas para solucionar um problema computacional

1. Definição do problema:

- quais são as informações disponíveis
- o que se deseja saber

2. Análise do problema:

- determinar o modelo de resolução
- selecionar o método
- construir o algoritmo

Obs.: Pode existir mais de um caminho para resolver o mesmo problema.

3. Programação, que de forma geral requer os seguintes passos:

- **Fluxograma**
  - Deve retratar, fielmente, o algoritmo escolhido.
  - Deve esclarecer os detalhes relacionados ao programa, independentes de linguagem de programação
  - Facilita o acompanhamento dos passos a serem seguidos para a solução do problema (pelo próprio programador ou usuário final)
  - Facilita o programador na fase de codificação.

- **Codificação**
  - É a escrita do programa usando as regras gramaticais de uma linguagem de programação.
  - Declaração dos tipos de dados.
  - Designação de memória para armazenamento de informações.
  - Especificação de formatos para os dados de entrada e saída.
  - Verificação das bibliotecas disponíveis que possam ser utilizadas.

- **Compilação**
  - É feito pelo próprio computador, consiste em traduzir o programa-fonte em programa-objeto.
  - É durante esse processo que o compilador detecta erros de sintaxe da linguagem, indicando o local do erro e diagnosticando a sua causa mais provável.

- **Verificação de erros de sintaxe**
  - Verificação, localização e remoção dos erros sintáticos detectados.
  - Se houver erros de sintaxe, o compilador não gera o programa-objeto.
  - Assim sendo, os erros devem ser corrigidos no programa-fonte que deve ser novamente compilado.

- **Link-edição**
  - Tendo todos os programas-fonte compilados deve-se juntá-los com as bibliotecas de funções necessárias para a resolução problema.
  - Esta etapa é feita pelo próprio computador através de comandos específicos da linguagem de programação.
  - O resultado da link-edição é um programa executável.

4. Preparação dos dados de entrada

Os dados de entrada devem ser preparados de acordo com os formatos especificados no programa-fonte.

5. Execução

O passo seguinte é executar o programa fornecendo os dados de entrada a fim de se obter os resultados do processamento.

6. Análise dos resultados

A interpretação dos resultados produzidos pelo computador para se assegurar que o problema foi corretamente resolvido.

É nesta etapa que se detecta os erros de lógica.

7. Relatório do programa

Para que um programa possa ser aceito como completo, o programador deve elaborar sua documentação, que consiste num relatório composto dos alguns itens principais:

- **Identificação**: onde deve constar o nome do programa, o nome do programador, a instituição a qual pertence e a data de programação.

- **Finalidade**: especificar o propósito do programa.

- **Modelo de resolução**: descrição do algoritmo ou método usado no programa, ou a citação de referências bibliográficas onde podem ser encontrados.

- **Restrições do programa,** onde devem constar
    - o intervalo de abrangência do programa,
    - os dimensionamentos de matrizes e vetores,
    - ocupação do espaço de memória para o programa,
    - estimativa de tempo do processamento para um problema típico,
    - nomes e detalhes dos arquivos usados,
    - subprogramas necessários, etc.

- **Tabela de variáveis**: apresentado as variáveis usadas no modelo de resolução e as correspondentes variáveis usadas no programa.

- **Modo de uso:** fornecendo informações sobre os dados de entrada (formatos, meios de registros, como os dados devem ser preparados) e sobre os resultados de saída.