Lógica de Programação

Para que se entenda sobre a lógica de programação, é fundamental saber sobre a sequência lógica, que nada mais é do que um grupo de passos estabelecidos para chegar a um objetivo ou solução. Trazem as instruções que devem ser feitas e a ordem que devem ser executadas, sendo as instruções comandos que indica o computador o que deve ser feito.

Um programa se classifica como uma sequência lógica de instruções organizadas para manipular informações inseridas pelos usuários.

Padronização de sequências lógicas:

Assim como os arquitetos elaboram plantas antes de desenhar ou renderizar algum projeto, os desenvolvedores de aplicativos precisam elaborar padronizações bem como as plantas arquitetônicas para que sejam compreensíveis para todos.

Os desenvolvedores buscam soluções lógicas para resolver problemas do cliente e, por isso, elaboram algoritmos que precisam estar em regras e padrões importantes para que possam ser entendidos para todos.

Uma linguagem em algoritmo necessita ser: linguagem clara e concisa; encadeamento lógico de instruções e objetivo claro a ser alcançado. No contexto da informática, é possível registrar algoritmos de três formas diferentes:

- Descrição narrativa: é como uma receita culinária porque precisa descrever o que deve ser feito na ordem correta. É também a mais próxima da nossa linguagem;
- Diagrama de blocos ou Fluxograma: é uma técnica intermediária que usa formas geométricas e frases curtas para descrever um processamento. É um método padronizado que

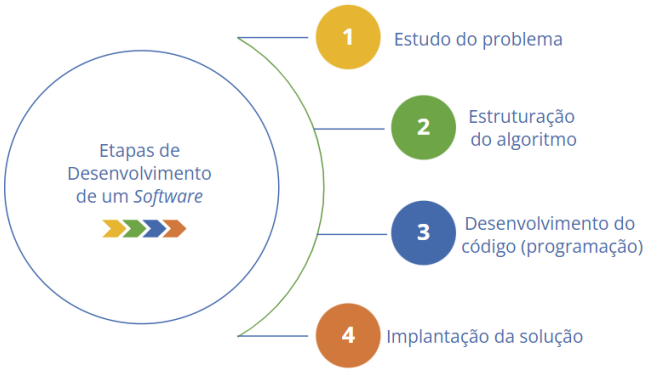
torna possível esquematizar o algoritmo definindo a entrada, processamento e saída de dados;

- Português estruturado ou pseudocódigo: forma mais detalhada e mais próxima dos códigos de programação. Todavia, gasta mais tempo e ganha-se tempo na hora de codificar um programa em uma linguagem específica.

É importante salientar que cada modo é aplicável de acordo com o nível de complexidade do projeto requerido.

o algoritmo tem começo, meio, fim e um objetivo a ser alcançado. Em outras palavras, o algoritmo deve ser visto como um projeto do programa, e seu desenvolvimento tem por foco completar uma missão de maneira inteligente, lógica e eficaz. Portanto, não é a solução e sim o caminho até ela de forma detalhada e manipulada de forma finita.

Observe as etapas necessárias para o desenvolvimento de um programa (*software*):



1: Estudo do problema:

Análise do problema e de seu contexto, para identificar a situação, refletir sobre suas características e apresentar as possíveis soluções

2: Estruturação do algoritmo

Escolha da melhor solução possível para resolver o problema e estruturação da proposta em forma de "projeto de programa" (algoritmo).

3: Desenvolvimento do código (programação)

Escrita do algoritmo a partir das regras e dos padrões de uma linguagem de programação específica, tal como C#, Java, C, C++, VB, PHP. É nesta etapa que ocorre a codificação do algoritmo.

4: Implantação da solução

Implantação do programa em situação de vida real, para verificar se o problema foi resolvido, e realização de eventuais ajustes (manutenção).

No uso do diagrama de blocos ou fluxograma é extremamente importante conhecer os símbolos utilizados nos gráficos:

São os tipos de dados:

Dados literais – ou caracteres – são sequências contendo letras, números e outros símbolos especiais. Uma sequência de caracteres deve ser

indicada entre aspas (“”). Esse tipo de dado é conhecido também como Alfanumérico, String, Literal ou Cadeia. Como exemplos, temos: “Fundação Bradesco”, “Técnico em Desenvolvimento de Sistemas”, “84”, “843.48”, entre outros.

São usados, por exemplo, os **comandos Leia** e **Escreva** para inserção (captura) e exibição (apresentação) de dados, respectivamente.

Além dos comandos, o pseudocódigo permite a declaração de **variáveis** e **expressões aritméticas** para realizar contas.

Os dados numéricos inteiros são definidos como tipos inteiros. Podem ser dados numéricos positivos ou negativos. Nesse tipo, não se encaixam números fracionários. Como exemplo, temos: 10, -10, 5, 85, -33, 88, -67, entre outros.

Regras na utilização dos pseudocódigos:

1. Todo algoritmo em pseudocódigo deve ser iniciado com **Algoritmo**: **NomeDoAlgoritmo**.
2. O início e fim do programa são limitados pelos marcadores **Início** e **Fim**.
3. As variáveis são declaradas no início do algoritmo, abaixo do marcador **Var**, da seguinte forma: **NomeDaVariável**: Tipo da variável.

Os nomes das variáveis **NÃO** podem:

Iniciar por número (erro: 1nome)

Ter espaço (erro: nome completo)

Ter caracteres especiais (´, `, ~,ç ,- e outros).

4. As palavras-reservadas devem ser evitadas: **Início**, **Fim**, **Var**, **Se** e **Senão**
5. Os nomes das variáveis são *case sensitive*. Dessa forma, ao manipularmos variáveis, devemos usar o mesmo nome declarado no início, considerando o uso de letras maiúsculas e minúsculas.

6. O comando **Leia** deve ser usado para receber (capturar) dados do usuário, fase do processamento conhecida como “Entrada de Dados”.
7. O comando **Escreva** deve ser usado para exibir (apresentar, mostrar) dados ao usuário, fase do processamento conhecida como “Saída de Dados”.
8. Os textos a serem exibidos na tela ou que devam ser inseridos como caractere são colocados entre "aspas“ (representação universal de um valor literal ou *string* – “Sistemas”).
9. Os comentários sobre o código podem ser inseridos {entre chaves} (incomum por confundir com agrupamentos) ou inseridos utilizando // (mais comum) no início da linha de instruções. O comentário não altera a execução do código. Contudo, ele é de fundamental importância para documentar e tornar inteligíveis as escolhas de programação realizadas no código para outros programadores.

São usados, por exemplo, os comandos **Leia** e **Escreva** para inserção (captura) e exibição (apresentação) de dados, respectivamente.

Além dos comandos, o pseudocódigo permite a declaração de variáveis e expressões aritméticas para realizar contas.

Observe esse algoritmo:

Algoritmo Conversao_real_dolar
// rotina que converte um determinado valor em reais em dólares
VALOR_REAL, COTACAO_DOLAR, VALOR_DOLAR: real
VALOR_REAL = 0
COTACAO_DOLAR = 0
VALOR_DOLAR = 0
Início
Escreva “Programa para converter reais em dólares”
Escreva “Informe o valor disponível em reais (para comprar dólares): R$”
Leia VALOR_REAL

Escreva “Informe o valor de cotação do dólar do dia: R$”
VALOR_DOLAR = VALOR_REAL / COTACAO_DOLAR
Escreva “Com essa quantia será possível comprar: US$”
Escreva VALOR_DOLAR
Escreva “Boa viagem!”
Fim

O que há de errado com esse algoritmo?

A variável COTACAO_DOLAR possui atribuição de valor igual a zero (COTACAO_DOLAR = 0). Após a inicialização, não há captura de valor de COTACAO_DOLAR atualizado pelo usuário. Dessa forma, o valor sempre será zero. Após a linha *Escreva “Informe o valor cotação do dólar do dia): R$”* está faltando o comando Leia COTACAO_DOLAR.

Controle de Fluxo e Estruturas de Repetição

Estruturas Sequenciais:

A estrutura sequencial é uma sequência de instruções que acontecem uma após a outra, sem desvios ou interrupções.

Esse tipo de estrutura é composto somente de linhas de comando sucessivas (que compõe um único fluxo possível de processamento) e está sempre limitado pelos marcadores INÍCIO e FIM do algoritmo ou de uma outra estrutura.

Estruturas de Seleção:

As estruturas de seleção são usadas para que sejam estabelecidos caminhos diferentes de instruções, a serem percorridos a partir de tomadas de decisão. Justamente por isso, esses recursos podem ser chamados de estruturas de seleção ou estruturas de decisão.

Como programador, você deverá utilizar os recursos de estruturas de seleção sempre que tiver de estruturar sequências de ações que poderão ser executadas ou não, a depender de um resultado frente a uma ou mais condições.

Existem três tipos de estruturas de seleção, cuja aplicação irá depender do contexto de utilização.

- Simples: levam os marcadores: **SE, ENTÃO e FIM SE**

  Uso: recurso a ser empregado em situações em que se faz necessário testar uma única condição/variável que, se verdadeira, irá desencadear a realização de um ou mais comandos. Temos, então, um teste e um grupo de ações que só acontecerão se a resposta for verdadeira. (não alteram o fluxo, só acrescentam).
- Composta: levam os marcadores: **SE, ENTÃO, SENÃO e FIM SE**

  Uso: recurso a ser empregado em situações em que se faz necessário testar uma única condição/variável que, se verdadeira, irá desencadear a realização de um ou mais comandos e que, se for falsa, irá desencadear outro grupo de ações. Temos então um teste e dois grupos de ações possíveis; um que acontecerá se a condição for verdadeira, e outro que acontecerá se a condição for falsa.
- Múltipla: Recursos a serem empregados em situações em que se faz necessário testar várias vezes a condição/variável. O resultado de cada teste irá desencadear um determinado grupo de ações.

  Utilizando recurso "Se" de forma encadeada (um dentro do outro). Marcadores: **SE, ENTÃO, SENÃO e FIM SE**

- Utilizando recurso "Faça Caso"
  Marcadores: **FAÇA CASO, CASO, OUTRO CASO, FIM CASO.**

Estruturas de Seleção Simples

A estrutura de seleção simples faz uso da instrução SE (IF). Ela é utilizada quando queremos testar uma condição antes de executarmos uma ou mais instruções.

Nessa estrutura, somente teremos uma ação SE o resultado da condição for verdadeiro. Desse modo, não há ação a ser executada caso o resultado seja falso.

Estruturas de Seleção Composta:

Na estrutura de seleção composta existem dois caminhos diferentes predeterminados.

Dessa maneira, haverá a execução de um comando ou grupo de comandos caso o resultado da condição seja verdadeiro, OU a execução de outro comando ou grupo de comandos diferente caso o resultado da condição seja falso. Ou seja, **independem entre si.**

Estruturas de Repetição:

Para o uso correto de estruturas de repetição, um item fundamental que deve ser compreendido é o controle do número de vezes que a estrutura será repetida. Desse modo, as repetições podem ser controladas a partir de alguns recursos.

Contador fixo:

Recurso que utiliza uma variável *Contador* para contar o número de vezes que uma determinada sequência será repetida. Quando o valor do contador atingir o número predeterminado, a repetição será encerrada.

Flag de resposta (sinalização do usuário):

Recurso que, ao final do processamento, pergunta ao usuário se ele deseja ou não executar a rotina de novo, armazenando o resultado da resposta em um *flag* de resposta, isto é, uma variável, por exemplo *Resp*.

Neste caso, o *loop* (ciclo ou laço) é interrompido pelo usuário.

Flag predeterminado (sinalização predeterminada):

Recurso que possibilita ao usuário encerrar o *loop* quando quiser, digitando uma palavra predeterminada, ou quando se chegar a uma condição específica, determinada pelo valor de um campo específico.