**Lógica de Programação**

Intervenção

Professor: João Vitor Rocha

***01- Programação, algoritmos e Conceitos Fundamentais***

Programação é o processo de escrita, teste e manutenção de um programa de computador. O programa é escrito em uma linguagem de programação, embora seja possível, com alguma dificuldade, o escrever diretamente em linguagem de máquina. Diferentes partes de um programa podem ser escritas em diferentes linguagens.

Diferentes linguagens de programação funcionam de diferentes modos. Por esse motivo, os programadores podem criar programas muito diferentes para diferentes linguagens; muito embora, teoricamente, a maioria das linguagens possa ser usada para criar qualquer programa.

Há várias décadas se debate se a programação é mais semelhante a uma arte (Donald Knuth), a uma ciência, à matemática (Edsger Dijkstra), à engenharia (David Parnas), ou se é um campo completamente novo.

Algoritmo é uma sequência lógica finita de passos para realizar uma tarefa ou resolver um problema. Em nosso dia a dia utilizamos algoritmos para realizar nossas atividades, definindo a sequência de atividades que devemos fazer para atingir um objetivo. Um exemplo simples é uma receita. Um algoritmo é, num certo sentido, um programa abstrato — dizendo de outra forma, um programa é um algoritmo concretizado. Os programas são visualizados mais facilmente como uma coleção de algoritmos menores combinados de um modo único — da mesma forma que uma casa é construída a partir de componentes.[1]

Dessa forma, um algoritmo é uma descrição passo a passo de como o computador irá executar uma operação específica, como, por exemplo, uma ordenação. Um programa, por outro lado, é uma entidade que na verdade implementa uma ou mais operações de forma que seja útil para as pessoas que o utilizam.

02-

**Identificadores**:Os "identificadores" ou "símbolos" são os nomes que você fornece para variáveis, tipos, funções e rótulos em seu programa. Os nomes de identificadores devem ser diferentes na ortografia e nas maiúsculas e minúsculas de todas as palavras-chave

**Variáveis**:Uma variável representa um contêiner ou espaço na memória física ou virtual de um computador, onde diferentes tipos de dados (**valores**) são armazenados durante a execução de um programa. A cada variável é atribuído um nome descritivo ou identificador que se refere ao valor salvo. Os dados armazenados podem mudar de valor ou serem constantes.

As variáveis são usadas para salvar e recuperar dados, representar valores existentes e atribuir novos. Elas permitem que os programadores se refiram a um valor com um nome representativo em vez de ter que se lembrar desse valor, o que facilita muitas tarefas complexas.

**Escopo**:Em linguagem de programação, o termo escopo refere-se à região de um programa onde uma determinada variável é acessível. O escopo determina a visibilidade e a disponibilidade de uma variável em diferentes partes do código.

**Tipos de Dados:**

Existem diversos tipos de dados utilizados na lógica de programação, cada um com suas características e finalidades específicas.

Aqui estão alguns dos principais tipos de dados:

* Números inteiros
* Números de ponto flutuante
* Booleanos
* Caracteres
* Strings

**03-** **Controles de Fluxos: Estruturas condicionais e de repetição;**  Controle de fluxo é a habilidade de ajustar a maneira como um programa realiza suas tarefas. Por meio de instruções especiais, chamadas comandos, essas tarefas podem ser executadas seletivamente, repetidamente ou excepcionalmente

Estruturas Condicionais:Quando falamos em lógica de programação, as estruturas condicionais são recursos oferecidos pelas linguagens para que seja possível verificar uma condição e alterar o fluxo de execução do algoritmo. Assim, é possível definir uma ação específica para diferentes cenários e obter exatamente o resultado esperado durante o desenvolvimento de um site ou de uma aplicação.

O uso das estruturas condicionais é praticamente indispensável na maioria dos projetos, já que elas são capazes de realizar diferentes funções de forma prática. Elas permitem, por exemplo, controlar o conteúdo que será exibido, criar formulários dinâmicos, desenvolver mídias interativas e tornar páginas responsivas.

As estruturas condicionais estão entre os conceitos básicos das linguagens de programação. Ainda assim, conhecer a fundo suas aplicações e as diferentes possibilidades que ela oferece é uma excelente maneira de se tornar um profissional mais completo e se destacar em suas criações.

Estrutura de repetição

Permite que uma sequência de ações ao longo do algoritmo seja executada repetidamente, até que uma determinada condição de interrupção seja satisfeita. A condição de interrupção é representada por uma expressão lógica.

Exemplos Portugol: Para, Enquanto, Repita

**04-Procedimentos, Funções e Recursividade;**

As **funções** (**functions**), também conhecidas como sub-rotinas, são muito utilizadas em programação. Um dos grandes benefícios é não precisar copiar o código todas as vezes que precisar executar aquela operação, além de deixar a leitura do código mais intuitiva. No exemplo anterior, caso precisássemos descobrir a raiz quadrada de 10 números, bastaria chamar a **função** que calcula a raiz quadrada 10 vezes.

Os **procedimentos** (**procedures**) diferem das funções apenas por não retornarem resultado, imagine um procedimento que envia e-mail. Precisa retornar resultado? Nos algoritmos apresentados aqui no **{ Dicas de Programação }** nós já utilizamos muitos **procedimentos** sem perceber. Por exemplo, para ler o valor digitado por um usuário nós já utilizamos o procedimento **LEIA** e para mostrar um texto na tela nós utilizamos o procedimento **ESCREVA**.

**05-. Cadeia de Caracteres, Estruturas unidimensionais e bidimensionais;**

Na programação de computadores, uma cadeia de caracteres ou string é uma sequência de caracteres, geralmente utilizada para representar palavras, frases ou textos de um programa.

As estruturas unidimensionais, conhecidas como vetores, consistem em um arranjo de elementos armazenados na memória principal, sequencialmente, todos com o mesmo nome (LOPES; GARCIA, 2002). Abaixo temos um exemplo de matriz corretamente declarado dentro da sintaxe adotada no livro da disciplina.

Matrizes: Os vetores bidimensionais ou matrizes são estruturas de dados que representam um conjunto de valores do mesmo tipo (estrutura homogênea), referenciáveis pelo mesmo nome e individualizados entre si através de sua posição de linha e coluna dentro desse conjunto (variáveis indexadas bidimensionais).

**06-Tipos Estruturados e Manipulação de Arquivos;**

Os dados estruturados são um formato padronizado para fornecer informações sobre uma página e classificar o conteúdo dela. Por exemplo, em uma página de receitas, quais são os ingredientes, o tempo e a temperatura de cozimento, as calorias e assim por diante.

**Exemplos de dados estruturados**

* Arquivos Excel.
* Bancos de **dados** SQL.
* **Dados** de pontos de vendas.
* Resultados de formulários da Web.
* Tags para Search Engine Optimization (SEO – Otimização de mecanismos de busca)
* Diretórios de produtos.
* Controle de inventário.
* Sistemas de reserva.

Criar pastas e manipular arquivos são operações com as quais todo usuário deve estar familiarizado. Algumas formas de manipular arquivos são: copiar, colar, excluir, mover, renomear e localizar arquivos e desfazer ações.

**07- Apontadores e Alocação Dinâmica;**

Em programação, um ponteiro ou apontador é um tipo de dado de uma linguagem de programação cujo valor se refere diretamente a um outro valor alocado em outra área da memória, através de seu endereço. Um ponteiro é uma simples implementação do tipo referência da Ciência da computação.

A alocação dinâmica permite ao programador criar variáveis e estruturas de dados em tempo de execução, ou seja, alocar memória para novo armazenamento de dado quando o programa está sendo executado.

**08-Depuração de código, versionamento e código limpo;**

Depuração corresponde ao processo de localizar e corrigir erros ou bugs no código-fonte de qualquer software. Quando o software não funciona conforme o esperado, os programadores de computadores estudam o código para determinar as causas dos erros.

O versionamento consiste em estratégias para gerenciar as diferentes versões de um código, de um sistema ou de um modelo. É uma forma de administrar as mudanças que são feitas e de garantir mais segurança na transição de uma versão para outra.

Em sua essência, Clean Code refere-se a um **estilo de programação que enfatiza a legibilidade**, a facilidade de manutenção e a eficiência do código.

Um código limpo é aquele que é fácil de entender e até mesmo modificar. Isso facilita a colaboração entre membros da equipe e reduz a incidência de problemas (bugs).

**09-Paradigmas de programação;**

Os paradigmas de programação são basicamente formas de abordar e resolver problemas ao escrever código. Eles são como conjuntos de regras que guiam os desenvolvedores. Cada linguagem de programação é criada com base nessas regras. Cada uma tem seu próprio conjunto de paradigmas que os desenvolvedores seguem ao escrever o código.

Esses paradigmas também definem o que não deve ser feito no código de um programa. Um renomado consultor de desenvolvimento chamado Robert Martin, conhecido como Tio Bob, criou esse termo. Para ele, paradigmas são como limites nas formas como as linguagens são usadas, orientando um estilo particular de programação. Por exemplo, na programação processual, não é recomendado pular desordenadamente pelo código fonte, e na programação funcional, não se deve modificar diretamente a memória do computador.

Os paradigmas de programação ajudam a estabelecer padrões para escrever código. Quando um especialista muda para uma linguagem de programação que segue um paradigma com o qual está familiarizado, é mais fácil para ele superar as diferenças entre as linguagens.

Existem dois principais paradigmas: o imperativo e o declarativo. Cada um deles inclui subtipos. O imperativo abrange a programação estruturada e a orientada a objetos (OOP), enquanto o declarativo engloba a programação funcional e lógica.

Agora que você já entendeu o que são os paradigmas de programação, chegou a hora de conhecer mais sobre os tipos existentes.

## **Paradigma Imperativo**

O paradigma imperativo é uma forma de programar que se concentra em dar instruções claras sobre como realizar uma tarefa. Para você entender melhor, é como se fosse uma receita de cozinha, onde você segue passos específicos para preparar aquele prato de doce ou salgado que você gosta.

Neste estilo de programação, podemos mudar o valor das coisas conforme precisarmos, como ajustar o volume de uma TV. Usamos comandos simples, como "atribuir um valor a uma variável" ou "fazer uma conta matemática". Também podemos tomar decisões (como "se isso acontecer, faça aquilo") e repetir ações várias vezes (como em um loop).

Além disso, podemos organizar nossos comandos em funções, que são como passos reutilizáveis em uma receita. Precisamos ter atenção à ordem em que os comandos são dados, pois ela faz diferença no resultado final. Podemos dividir nossa tarefa em pequenas etapas para facilitar a compreensão e organização.

Às vezes, usamos objetos para representar coisas do mundo real, como em jogos ou programas mais complexos. O paradigma imperativo é usado em muitas linguagens de programação populares e é ótimo para tarefas práticas, como lidar com dados e interagir com sistemas.

## **Paradigma estruturado**

Também conhecida como programação estruturada, o paradigma estruturado é um estilo dentro do paradigma imperativo que se concentra em organizar o código fonte em uma sequência de procedimentos ou funções. Cada procedimento é uma série de comandos que realiza uma tarefa específica. Algumas das linguagens de programação utilizadas são: C, Pascal, COBOL, ALGOL, BASIC, Fortran.

Para você entender melhor, vamos apresentar um exemplo usando a Programação Estruturada para escrever um programa que guia o cozinheiro na preparação da sopa de cebola.

**Procedimento**: Cortar Cebolas

**Descrição**: Este procedimento trata de cortar as cebolas em pedaços pequenos.

**Procedimento**: Refogar cebolas

**Descrição**: Este procedimento aborda o processo de refogar as cebolas cortadas.

**Procedimento**: Adicionar caldo e cozinhar

**Descrição**: Este procedimento lida com a etapa de adicionar o caldo à sopa e cozinhar até que esteja pronta.

**Procedimento**: Servir

**Descrição**: Este procedimento finaliza a sopa, preparando-a para ser servida.

Ao chamar esses procedimentos na ordem correta, você terá um programa que guia o cozinheiro passo a passo na preparação da sopa de cebola. Por exemplo:

**Cortar cebolas**

**Refogar cebolas**

**Adicionar caldo e cozinhar**

**Servir**

Isso representa uma sequência clara de etapas para preparar a sopa de cebola. Cada procedimento representa uma etapa específica no processo de preparação da receita.

## **Paradigma orientado a objetos**

O paradigma orientado a objetos é um paradigma de programação centrado na ideia de 'objetos'. Cada objeto, uma instância de uma classe, encapsula dados (atributos) e comportamentos (métodos), permitindo interações entre eles para executar tarefas. Essa abordagem facilita a modelagem de sistemas complexos ao representar entidades do mundo real com características e ações específicas.

O paradigma orientado a objetos também inclui conceitos-chave como encapsulamento, onde dados e métodos são agrupados em objetos, limitando o acesso externo. Além disso, a herança possibilita que classes herdem características de outras, promovendo a reutilização de código fonte e hierarquias de classes. Polimorfismo permite que métodos tenham diferentes implementações em diversas classes, proporcionando flexibilidade ao adaptar o código a diferentes situações. A associatividade entre classes e políticas de acesso (público, privado, protegido) complementam as características fundamentais do paradigma orientado a objetos, tornando-a uma ferramenta importante para desenvolvimento de software.

## **Paradigma Declarativo**

O paradigma declarativo é uma um paradigma de programação que foca em descrever o que deve ser alcançado, em vez de como alcançá-lo. Em vez de fornecer uma sequência de instruções detalhadas, o programador declara as propriedades e relações que o resultado final deve ter.

Para ilustrar, continuaremos usando o exemplo da sopa de cebola.

**Objetivo**: Produzir uma sopa de cebola deliciosa.

**Descrição Declarativa:** A sopa deve ter uma base rica e saborosa, com cebolas caramelizadas e caldo de carne bem temperado. Além disso, deve ser gratinada no forno com queijo derretido por cima até dourar.

**Passos:**

Caramelizar as cebolas até que fiquem douradas e adquiram um sabor adocicado.

Preparar um caldo de carne robusto com ervas aromáticas.

Misturar o caldo com as cebolas caramelizadas para criar a base da sopa.

Levar a sopa ao forno com uma camada generosa de queijo por cima, gratinando até que fique dourada e borbulhante.

## **Paradigma Funcional**

A programação funcional se baseia no uso de funções como peças fundamentais do código. Ela permite que as funções sejam tratadas de maneira flexível, como blocos de construção que podem ser usados, passados e armazenados em diferentes partes do programa. Uma característica crucial é a imutabilidade dos dados, o que significa que uma vez criados, eles não podem ser alterados. Isso resulta em um código mais previsível e com menos efeitos colaterais indesejados. Essa abordagem se alinha ao paradigma declarativo, que foca em "o que deve ser feito" em vez de "como fazer", tornando o código mais claro e eficiente. A programação funcional, ao combinar a imutabilidade dos dados e a flexibilidade das funções, se destaca como uma opção poderosa para solucionar desafios complexos de forma elegante.

Além disso, a programação funcional utiliza a recursividade para abordar problemas de maneira direta e concisa. Ela também destaca o uso de funções de alta ordem, que podem aceitar ou retornar outras funções. Essa abordagem, fundamentada na teoria matemática das funções, é implementada em linguagens de programação como Haskell, Lisp, Clojure e integra partes de linguagens de programação multiparadigma como JavaScript e Python. Ao priorizar a simplicidade, a expressividade e a minimização de efeitos secundários, a programação funcional oferece uma abordagem altamente eficaz para resolver problemas.

## 

## **Paradigma lógico**

## Além do paradigma imperativo e declarativo, também há o paradigma lógico, um paradigma de programação que usa a lógica matemática para resolver problemas. Em vez de focar em sequências de comandos ou funções, como em outros paradigmas, o paradigma lógico utiliza regras lógicas para definir as relações entre os elementos do problema. Isso significa que o programador especifica as condições e relações que devem ser verdadeiras para obter o resultado desejado, e o sistema deduz a solução com base nessas regras. Prolog é uma das linguagens de programação que segue esse paradigma, e é amplamente usada em áreas como inteligência artificial e processamento de linguagem natural.

## O paradigma imperativo, por exemplo, foca em dar instruções específicas para executar uma tarefa, como seguir uma receita de cozinha. Já o paradigma de programação declarativo se concentra em descrever o que deve ser alcançado, sem entrar em detalhes sobre como fazer. O paradigma funcional coloca as funções como peças centrais do código, permitindo flexibilidade e imutabilidade dos dados. E o paradigma lógico se baseia em regras lógicas para deduzir soluções a partir das informações fornecidas.

**10-Noções de programação funcional e programação lóg**

A programação funcional é um paradigma de programação que se concentra na avaliação de expressões e na aplicação de funções matemáticas. Ela trata a computação como a avaliação de funções e evita o uso de estados mutáveis e dados compartilhados.

A Ciência de Dados utiliza a PF, sobretudo na análise e no processamento de dados, pois permite manipular as informações de forma concisa e declarativa, por meio de funções de ordem superior, como map, filter e reduce.

O paradigma também é ideal para algoritmos de busca e ordenação e para o processamento de eventos e reações.

A linguagem Haskell, por exemplo, é exclusivamente funcional, enquanto outras como C, C#, C++, PHP, Scala, Rust e JavaScript são multiparadigmas e também usam a abordagem da programação funcional.

Programação lógica é um paradigma de programação que faz uso da lógica matemática. John McCarthy [1958] foi o primeiro a publicar uma proposta de uso da lógica matemática para programação.

A primeira linguagem de programação lógica foi a Planner, a qual permitia a invocação orientada a padrões de planos procedimentais de asserções e de objetivos. Com a necessidade de adaptação aos sistemas de memória muito limitada, que eram disponíveis quando foi desenvolvida. A linguagem Planner usava estruturas de controle de backtracking, de tal forma que apenas um único caminho computacional tinha que ser armazenado por vez. Em seguida, o Prolog foi desenvolvido como uma simplificação do Planner que permitia a invocação orientada a padrões apenas a partir de objetivos (também baseado em backtracking).