Programação é para mim?

Autor: Daniel Naueji

Starprog00@gmail.com

Cont: 925948312

O que é Lógica de Programação?

Lógica de programação refere-se ao processo de pensar e criar algoritmos para resolver problemas computacionais. É a base para escrever códigos que as máquinas podem entender e executar. Envolve o uso de estruturas de controle (como loops e condicionais) e princípios matemáticos e lógicos para desenvolver soluções eficientes.

Para que serve?

A lógica de programação serve para:

1. **Desenvolver Algoritmos:** criar sequências de passos para resolver problemas específicos. Algoritmos são o núcleo de qualquer software ou aplicativo.

2. **Criar Código Eficiente:** utilizar princípios lógicos para escrever código que realiza tarefas de forma eficiente e sem erros.

3. **Automatizar Processos**: reduzir a necessidade de intervenção humana ao criar sistemas que executam tarefas automaticamente com base em dados e regras predefinidos.

4. **Resolver Problemas:** oferecer uma abordagem estruturada para resolver problemas complexos, dividindo-os em partes menores e mais gerenciáveis.

Importância no Mundo da Automação

A lógica de programação é crucial na automação porque:

1. **Automação de Processos Repetitivos:** permite criar scripts e programas que automatizam tarefas repetitivas e demoradas, aumentando a eficiência e reduzindo erros humanos.

2. **Desenvolvimento de Sistemas de Controle**: em indústrias, a lógica de programação é usada para criar sistemas que controlam maquinários e processos de produção de forma autônoma.

3. **Análise e Processamento de Dados:** automatiza a coleta, análise e processamento de grandes volumes de dados, gerando relatórios e insights valiosos.

4. **Integração de Sistemas:** facilita a comunicação entre diferentes sistemas e plataformas, automatizando a troca de dados e a execução de processos complexos.

Aplicações da Lógica de Programação

1. **Desenvolvimento de Software:** usada para criar aplicações, sistemas operacionais e programas de software.

2. **Desenvolvimento Web:** para criar e manter sites e aplicativos web, utilizando linguagens como HTML, CSS, JavaScript, PHP, etc.

3. **Automação de Testes**: em engenharia de software, para criar testes automatizados que verificam a funcionalidade de aplicativos e sistemas.

4. **Inteligência Artificial e Machine Learning:** Algoritmos complexos e lógicos são a base para o desenvolvimento de modelos que aprendem e tomam decisões com base em dados.

5. **Sistemas Embarcados:** em dispositivos como microcontroladores e sistemas embarcados, a lógica de programação é usada para controlar dispositivos e sensores.

6. **Financeiras e Comerciais:** automatiza operações financeiras, controle de inventário, e sistemas de recomendação em plataformas de ecommerce.

7. **Educação:** Ferramentas educacionais e simuladores são desenvolvidos para ensinar e aprender conceitos de lógica e programação.

Conceitos Fundamentais

Para entender a lógica de programação, você deve estar familiarizado com:

- **Algoritmos:** um algoritmo é uma sequência finita de instruções bem definidas que, partindo de um estado inicial, passam por uma série de estados sucessivos e terminam em um estado final. Obtendo uma solução para um problema especifico.

Importância: é fundamental na programação, pois cada programa que você escreve é, na essência, um algoritmo que o computador segue para realizar uma tarefa.

Característica de um algoritmo:

1. **Claridade:** as instruções devem ser claras e precisas, sem ambiguidade.
2. **Eficiência:** deve usar os menores recursos possíveis, como o tempo e memória.
3. **Finitude:** um algoritmo deve ter um número finito de passos e terminar em algum momento.

Exemplo de alguns algoritmos no dia-a-dia:

Fazer um café:

1. Ferver a água.
2. Coloque o pó do café no filtro.
3. Despeje a água quente sobre o pó.
4. Espere o café coar.
5. Sirva o café.

- **Estruturas de Controle:** estruturas de controle são fundamentais na lógica de programação. Elas permitem que um programa tome decisões e execute diferentes blocos de código dependendo das condições específicas. Existem três tipos principais de estruturas de controle:

1. **Estruturas de Controle Condicionais:**

As estruturas condicionais permitem que o programa tome decisões e execute diferentes seções de código com base em condições específicas. As principais estruturas condicionais são:

SE condição ENTÃO

// Código a ser executado se a condição for verdadeira

SENÃO

// Código a ser executado se a condição for falsa

FIM SE

2. **Estruturas de Controle de Repetição:**

Essas estruturas permitem que um bloco de código seja executado repetidamente, dependendo de uma condição ou por um número específico de vezes. As principais estruturas de repetição são:

ENQUANTO condição FAÇA

// Código a ser executado enquanto a condição for verdadeira

FIM ENQUANTO **- Estruturas de Dados:**

Estruturas de dados são maneiras específicas de organizar e armazenar dados em um computador para que possam ser acessados e manipulados de forma

eficiente. Elas são fundamentais em ciência da computação e programação porque influenciam a eficiência dos algoritmos e o desempenho geral dos programas.

Objetivo das Estruturas de Dados

1. **Organização:** organizar dados de maneira que seja fácil de acessar e manipular.

2. **Eficiência:** permitir que operações como inserção, exclusão, e busca sejam realizadas de forma eficiente.

3. **Gerenciamento:** facilitar o gerenciamento da memória e o controle de dados complexos.

Tipos Comuns de Estruturas de Dados

1. Variáveis Simples

- **Descrição:** armazenam valores únicos.

- **Exemplo:** inteiros, floats, strings.

- **Funções e Procedimentos:**

Funções e procedimentos são conceitos fundamentais em programação e em pseudocódigo. Eles ajudam a organizar o código, promovem a reutilização e facilitam a manutenção. Embora os termos sejam usados de maneira diferente em várias linguagens de programação, eles têm conceitos semelhantes.

Funções: são blocos de código que realizam uma tarefa específica e retornam um valor. Elas podem receber parâmetros de entrada e usar esses parâmetros para calcular um valor de saída.

Características:

1. Entrada (Parâmetros): Funções podem receber zero ou mais parâmetros que fornecem os dados necessários para o processamento.

2.Saída (Valor de Retorno): Funções retornam um valor ao chamador.

3. Reusabilidade: Funções podem ser chamadas várias vezes em diferentes partes do programa.

Sintaxe de Pseudocódigo:

```pseudocode
FUNÇÃO NomeDaFuncao(parâmetros)
    // Código para realizar a tarefa
retornar valor
FIM FUNÇÃO
```

Exemplo:

```pseudocode
FUNÇÃO Soma(a,
b)
    resultado ← a + b
retornar resultado
FIM FUNÇÃO

// Uso da função
total ← Soma(5, 3) // total será 8
```

Quando Usar Funções:

- Quando você precisa calcular e retornar um valor.
- Quando deseja modularizar o código para facilitar a leitura e a manutenção.

Procedimentos: são blocos de código que realizam uma tarefa específica, mas não retornam um valor. Eles podem modificar variáveis globais ou realizar ações como exibir mensagens.

Características:

1. **Entrada (Parâmetros):** Procedimentos podem receber zero ou mais parâmetros para influenciar sua execução.
2. **Sem Saída (Valor de Retorno):** Procedimentos não retornam um valor ao chamador; em vez disso, realizam ações diretamente.
3. **Reusabilidade**: Procedimentos podem ser chamados várias vezes para executar uma série de instruções.

Sintaxe de Pseudocódigo:

```pseudocode
PROCEDIMENTO NomeDoProcedimento(parâmetros)
    // Código para realizar a tarefa
FIM PROCEDIMENTO
```

Exemplo:

```pseudocode
PROCEDIMENTO ImprimirMensagem(mensagem)
    IMPRIMIR mensagem
FIM PROCEDIMENTO

// Uso do procedimento
ImprimirMensagem("Olá, Mundo!") // Imprime "Olá, Mundo!"
```

Quando Usar Procedimentos:

- Quando você deseja realizar uma tarefa sem retornar um valor.
- Quando você quer modificar o estado global ou realizar ações como exibir dados.

- **Pseudocódigo e Diagramas de Fluxo:** são ferramentas comuns usadas no desenvolvimento de algoritmos para descrever a lógica de programação de forma simples e visual.

Pseudocódigo:

- É uma forma de descrever algoritmos usando uma linguagem intermediária entre a linguagem natural e a linguagem de programação.

- Não segue a sintaxe rigorosa de nenhuma linguagem de programação, mas utiliza uma estrutura lógica que facilita a compreensão do algoritmo.

- Seu objetivo é ser claro suficiente para ser entendido por pessoas, independentemente de conhecimentos técnicos, e, ao mesmo tempo, detalhado o suficiente para que possa ser facilmente em código.

Exemplo:

Inicio

Leia A, B

Se A > B então

Exiba “A é maior que B”

Senão

Exiba “B é maior que A”

Fim

Diagrama de fluxo (Fluxograma)

- Representa graficamente um algoritmo, usando símbolos padronizados para cada tipo de operação, como decisões, ações ou entrada/saída de dados.

- Ajuda a visualizar o fluxo de execução do programa, facilitando a compreensão da sequência de etapas e decisões lógicas.

- Os principais símbolos incluem:

- **Elipse:** início/fim.
- **Retângulo:** ação ou processamento.
- **Losango:** decisão ou condição.
- **Paralelogramo:** entrada/saída de dados.

Obs: A figura só ilustraremos nos fascículos posteriores.

CONCLUINDO

A lógica de programação é essencial para a criação e desenvolvimento de soluções tecnológicas modernas. Ela é o alicerce para resolver problemas de forma eficiente e clara. Entender a lógica de programação é essencial para criar algoritmos que possam ser traduzidos para qualquer linguagem de programação. Dominar esse conceito permite:

1. **Pensamento estruturado:** desenvolver uma abordagem sistemática para resolver problemas, dividindo-os em passos menores.

2. **Clareza:** ao entender a lógica por trás dos algoritmos, o programador (você) consegue construir soluções que são mais fáceis de entender, modificar e depurar.

3. **Flexibilidade:** a lógica de programação não é específica de uma linguagem; uma vez compreendida, pode ser aplicada a diferentes linguagens e tecnologias.

4. **Solução de problemas:** a lógica permite identificar os passos críticos para chegar a uma solução, utilizando condições, repetições e sequências de maneira eficaz.

Portanto, dominar a lógica de programação é a chave para transformar ideias em códigos funcional e bem-estruturado.

Exercícios básicos

1 - Elabore um programa que mostra o nome e a idade do indivíduo.

2 – Faça um algoritmo que lê dos números, e que ilustra o maior dele e se for igual exibe na tela números **iguais.**

3 – Crie um programa que faz a soma de dois números.

4 – Elabore um algoritmo que mostra quantos municípios há na sua província.

5 – Faça um programa que mostra a média final de um estudante.

Obs: faça os exercícios no seu caderno e comece praticando criando alguns exercícios da sua autoria.

"Desafie-se a resolver problemas do mundo real"