******

Thauã Arruda Figueiredo

**(Algoritmos e lógica de programação)**

Cuiabá, 2024

Thauã Arruda Figueiredo

**(Algoritmos e lógica de programação)**

Trabalho apresentado no sesc escola (senac) , curso de jogos digitais, algoritmos e lógica de programação.

Orientador: Wanderson Timóteo

Cuiabá, 2024

**SUMÁRIO**

1. ALGORITMO
   1. DEFINIÇÃO DE ALGORITMOS E SUA IMPORTÂNCIA NA PROGRAMAÇÃO...........................................................................4
   2. LÓGICA DE PROGRAMAÇÃO E SUA RELAÇÃO COM ALGORITMOS...............................................................................4
2. ESTRUTURAS DE CONTROLE
   1. PESQUISA SOBRE ESTRUTURAS DE CONTROLE COMO SEQUÊNCIA, SELEÇÃO E REPETIÇÃO......................................4
   2. EXEMPLOS DE ALGORITMOS QUE UTILIZAM ESTRUTURAS DE CONTROLE.............................................................................4
3. TIPOS DE DADOS E VARIÁVEIS
   1. DEFINIÇÃO DE TIPOS DE DADOS BÁSICOS, COMO INTEIROS, PONTO FLUTUANTE, BOOLEANOS, ETC................4
   2. EXEMPLOS DE DECLARAÇÃO E UTILIZAÇÃO DE VARIÁVEIS EM ALGORTIMOS.........................................................................4
4. FUNÇÕES E MODULARIZAÇÃO
   1. PESQUISA SOBRE O CONCEITO DE FUNÇÃO NA PROGRAMAÇÃO...........................................................................4
   2. A IMPORTÂNCIA DA MODULARIZAÇÃO NA ESCRITA DE ALGORITMOS...............................................................................4
   3. EXEMPLOS DE FUNÇÕES SIMPLES E SUA UTILIZAÇÃO EM ALGORITMOS MAIS COMPLEXOS..............................................4

**1.0 ALGORITMO**

**1.1** DEFINIÇÃO DE ALGORITMOS E SUA IMPORTÂNCIA NA PROGRAMAÇÃO

Algoritmo

É uma forma de mostrar como concluir uma tarefa ou resolver um problema. Algoritmos são usados ​​em computação, processamento de dados e otimização em vários campos, como ciência da computação, matemática e engenharia. É importante no desenvolvimento de computadores e software porque ajuda a definir como os programas funcionam.

Importância na programação:

1. Resolução de problemas: Algoritmos são formas estruturadas de resolver problemas. Ajuda os desenvolvedores a dividir tarefas complexas em partes mais simples e gerenciáveis.

2. Início: Bons algoritmos são importantes para garantir um melhor desempenho do programa.

3. Reutilização de código: Algoritmos modulares bem definidos que facilitam a reutilização de código. Os moderadores podem economizar tempo e esforço usando algoritmos em outras partes do projeto ou em outros projetos.

4. Manutenção de código: Algoritmos simples e bem escritos que cuidam da manutenção de código.

5. Programação de aprendizagem: Algoritmos são uma parte importante da programação de aprendizagem.

6. Automação de tarefas: Algoritmos são usados ​​para automatizar processos repetitivos e enfadonhos.

7. Tomada de decisão: Algoritmos são usados ​​para tomar decisões com base em dados e critérios específicos. Isto é especialmente importante em áreas como a inteligência artificial, onde os algoritmos aprendem e melhoram ao longo do tempo.

**1.2** LÓGICA DE PROGRAMAÇÃO E SUA RELAÇÃO COM ALGORITMOS

Lógica de programação:

A lógica de programação está intimamente relacionada aos algoritmos e é uma estrutura teórica que orienta o desenvolvimento de algoritmos eficientes e precisos.

Sua relação com os algoritmos:

Perguntas de programação: É assim que os programadores pensam e criam soluções para problemas antes de traduzi-los para uma linguagem de produção.

Descrição: Implementação desta lógica, explicando passo a passo como funciona a solução.

Lógica do Programa: Envolve a compreensão das estruturas de controle de fluxo que controlam o caminho de execução de um programa, como restrições e loops (if-else).

Conteúdo: Use estruturas de controle para orientar o comportamento do programa em diferentes situações e contextos.

Programação: inclui a capacidade de organizar e manipular dados e compreender conceitos como variáveis, matrizes e estruturas de dados. Variáveis: Use estes conceitos para especificar e manipular dados durante a execução do programa.

Análise de Programa: Oferece a capacidade de analisar problemas, identificar áreas críticas e identificar estratégias de solução.

Algoritmo: apresenta detalhadamente a estratégia de solução, mostrando passo a passo como processar os dados e como resolver o problema.

Lógica associativa: Inclui tentar ser eficaz na resolução de problemas e esclarecer declarações lógicas.

Comparação: Os algoritmos devem ser apresentados de forma eficiente e clara e garantir que as soluções sejam compreendidas e implementadas.

Em suma, a lógica é um método teórico e teórico para resolver problemas, e os algoritmos são a principal expressão e implementação dessa lógica.

**2.0 ESTRUTURAS DE CONTROLE**

**2.1** PESQUISA SOBRE ESTRUTURAS DE CONTROLE COMO SEQUÊNCIA, SELEÇÃO E REPETIÇÃO

Estruturas de controle são elementos fundamentais da lógica de programação e são usadas para controlar o fluxo de programação. Estrutura sequencial: A estrutura sequencial é a forma mais simples de controle de fluxo e consiste em uma sequência linear de instruções que são executadas individualmente na ordem em que aparecem no código. Estruturas (ou métodos) de decisão: As estruturas de decisão permitem que um programa siga diferentes cursos de ação com base em condições ou expressões booleanas. Essas estruturas organizacionais podem ser combinadas e aninhadas de diversas maneiras para criar mudanças complexas que atendam a diferentes requisitos. Por exemplo, é comum usar estruturas de decisão em uma estrutura de repetição para controlar o comportamento do programa com base em diferentes condições durante a execução. Isso permite que os programadores controlem o fluxo da programação e tomem decisões com base em diferentes situações, resolvendo assim diversos problemas de software.

Os controles, também chamados de estruturas de processo, são importantes na programação de computadores. Ao usar comandos em um programa, os desenvolvedores podem controlar o fluxo de comandos de determinadas maneiras. As instruções são executadas individualmente e aparecem no código, sem exceções ou bifurcações no fluxo de trabalho. A estrutura é representada por uma lista sequencial de instruções, na qual cada linha de código é executada. Decisão (ou escolha): O processo de tomada de decisão permite que um programa execute diferentes ações com base em informações especificadas. Isso é feito usando cláusulas condicionais como if, if, etc. Calcular a idade do usuário Mostrar "Você é legal" Toque "Você ainda é jovem" Loops (ou Loops): Os loops são projetados para permitir que o código seja executado quando certas condições forem atendidas. Esses recursos podem ser combinados e implementados para criar estilos que atendam a diferentes necessidades. Os programadores podem controlar o fluxo do programa e tomar decisões com base em diferentes cenários para resolver diferentes problemas do computador.

**2.2** EXEMPLOS DE ALGORITMOS QUE UTILIZAM ESTRUTURAS DE CONTROLE

Aqui estão alguns exemplos simples de algoritmos usando distribuição, distribuição e iteração: A média de três anos é calculada como: Este é um exemplo de algoritmo sequencial, que executa instruções na ordem em que aparecem. Como verificar se é antigo ou novo: 4. Exiba "Este número é o mesmo" 6. Exibir "Quantidade é diferente" Neste algoritmo, um método de seleção (se-então) é usado para selecionar a mensagem a ser exibida com base no número de bits. Este algoritmo usa métodos de conversão (em loop) para representar números naturais de 1 a 10. Como calcular a soma dos números entre 1 e 100: (2). O número do índice começa em 0 (3). Comece a mudar (i) para 1 (3). Comece a mudar (i) para 17 Finalmente (7). fim do tempo. O algoritmo também usa estruturas temporárias para somar números entre 1 e 100. Estes são exemplos básicos que mostram como usar controle, seleção e interação em algoritmos simples.

**3.0** TIPOS DE DADOS E VARIÁVEIS

**3.1** DEFINIÇÃO DE TIPOS DE DADOS BÁSICOS, COMO INTEIROS, PONTO FLUTUANTE, BOOLEANOS, ETC

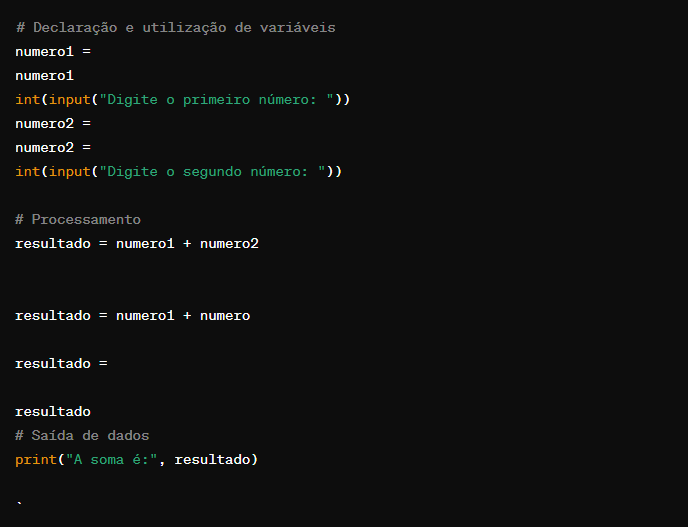
Os tipos de dados básicos são a base do projeto e são usados ​​para representar diferentes tipos de informações no projeto. Aqui estão as definições de tipos de dados comuns: Símbolos inteiros, ou seja, números sem decimais, representa verdade lógica (Verdadeiro) ou falso (Falso). Representantes dos acionistas pessoas físicas, representa uma lista de pontos (uma matriz de 8 bits usada para armazenar dados binários). A maioria das linguagens de programação possui um conceito de nulo para indicar a ausência de um valor de variável. Tipo de dados (objeto, elemento, dinâmico, etc.) Algumas linguagens fornecem tipos de dados longos que podem armazenar qualquer tipo de valor. Além disso, algumas linguagens de programação possuem tipos de dados adicionais e funções especiais.

**3.2** EXEMPLOS DE DECLARAÇÃO E UTILIZAÇÃO DE VARIÁVEIS EM ALGORTIMOS

Vamos criar alguns exemplos de declaração e utilização de variáveis em algoritmos usando a linguagem de programação Python, que é conhecida por sua sintaxe simples e legibilidade.



OU

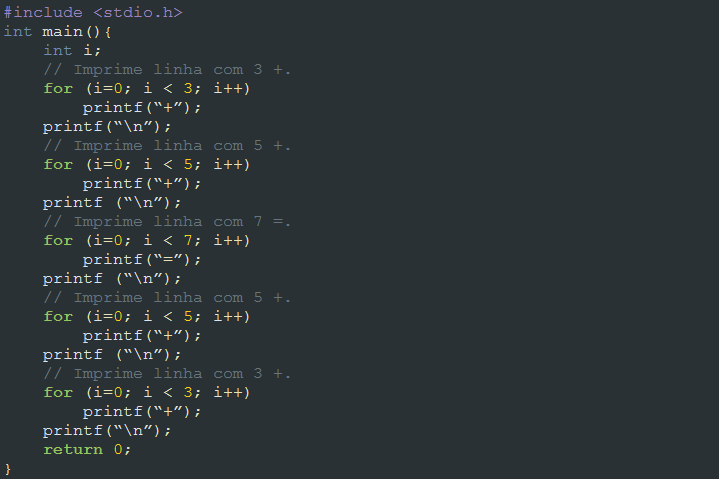


Estes exemplos são escritos em Python, uma linguagem de alto nível que é fácil de entender. Eles demonstram a declaração de variáveis, entrada de dados, processamento e saída de resultados. Lembre-se de que a sintaxe pode variar dependendo da linguagem de programação que você está utilizando.

**4.0** FUNÇÕES E MODULARIZAÇÃO

**4.1** PESQUISA SOBRE O CONCEITO DE FUNÇÃO NA PROGRAMAÇÃO

Em programação, uma função é uma sequência de instruções ou um bloco de código que realiza uma tarefa específica e pode ser chamado ou invocado a partir de outras partes do programa. As funções ajudam a organizar e modularizar o código, permitindo a reutilização de código e a separação de responsabilidades.



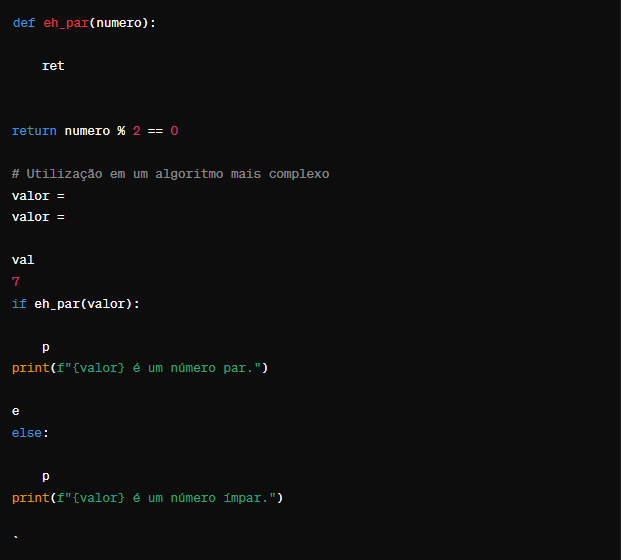
As funções são fundamentais para a estruturação e organização de código, promovendo a reutilização e modularidade. Elas facilitam o desenvolvimento e a manutenção de programas, especialmente em projetos mais extensos.

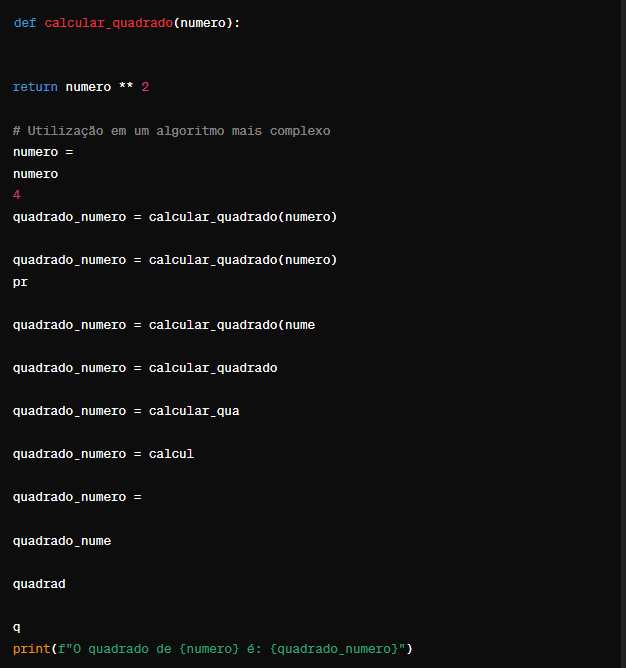
**4.2** A IMPORTÂNCIA DA MODULARIZAÇÃO NA ESCRITA DE ALGORITMOS

A modularização é um conceito importante ao escrever algoritmos que se referem à divisão de um programa complexo em partes menores, mais gerenciáveis ​​e independentes, chamadas módulos ou funções. Essa abordagem traz muitos benefícios importantes para o desenvolvimento de software.

Esses componentes podem ser reaproveitados em diferentes partes do programa ou em outros projetos, economizando tempo e esforço. Isso facilita a identificação e a correção de erros, pois alterações e modificações nos componentes podem ser feitas sem interromper todo o sistema. - Essa alteração facilita a leitura do código porque você pode inspecionar cada seção individualmente.

**4.3** EXEMPLOS DE FUNÇÕES SIMPLES E SUA UTILIZAÇÃO EM ALGORITMOS MAIS COMPLEXOS





Estes são exemplos simples que ilustram como funções podem ser definidas para tarefas específicas e como essas funções podem ser integradas em algoritmos mais complexos. A modularização proporcionada pelas funções melhora a legibilidade, reutilização e manutenção do código. À medida que os algoritmos se tornam mais elaborados, a utilização de funções é essencial para organizar o código de forma eficiente.