2024

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO

NÃO PRECISA

Aluno: MATHEO DE AMORIM ACOSTA

**Algoritmos e lógica de programação**

Definição de algoritmos e sua importância na programação.......................................1

Lógica de programação e sua relação com algoritmos................................................2

Pesquisa sobre estruturas de controle como sequência, seleção e repetição............3

Exemplos de algoritmos que utilizam estruturas de controle.......................................4

Definição de tipos de dados básicos, como inteiros, ponto flutuante, booleanos, etc.5

Exemplos de declaração e utilização de variáveis em algoritmos...............................6

Pesquisa sobre conceito de função na programação..................................................7

A importância da modularização na escrita de algoritmos..........................................8

Exemplos de funções simples e sua utilização em algoritmos mais complexos..........9

**ALGORITMOS**

Um algoritmo é uma sequência finita e ordenada de instruções bem definidas e não ambíguas que resolvem um problema ou realizam uma tarefa específica. Em outras palavras, é um conjunto de passos que, quando seguidos corretamente, levam à solução de um problema.

**Importância na Programação**

**Resolução de Problemas:** Algoritmos fornecem uma abordagem sistemática para resolver problemas, ajudando os programadores a dividir tarefas complexas em etapas mais gerenciáveis.

**Eficiência:** Algoritmos eficientes são cruciais para garantir que o software execute de forma rápida e eficaz, mesmo para grandes volumes de dados.

**Reusabilidade:** Algoritmos bem projetados podem ser reutilizados em diferentes contextos e em diferentes partes de um programa, economizando tempo e esforço de desenvolvimento.

**Manutenção**: Algoritmos claros e bem documentados facilitam a manutenção e o entendimento do código ao longo do tempo, mesmo por outros desenvolvedores.

**Lógica de Programação**

A lógica de programação é a capacidade de pensar de forma estruturada e sequencial para resolver problemas utilizando algoritmos. Envolve a aplicação de conceitos como condições, loops, estruturas de dados e operações lógicas para criar algoritmos eficazes.

**Relação com Algoritmos**

**Expressão de Algoritmos:** A lógica de programação fornece os blocos de construção necessários para expressar algoritmos de forma clara e precisa em uma linguagem que os computadores possam entender e executar.

**Organização de Pensamento:** Através da lógica de programação, os desenvolvedores aprendem a decompor problemas complexos em tarefas mais simples, identificando padrões e definindo estratégias para resolvê-los.

**Eficiência e Eficácia:** Uma boa compreensão da lógica de programação permite que os programadores criem algoritmos eficientes e eficazes, minimizando erros e maximizando o desempenho do software.

**Flexibilidade:** Com uma base sólida em lógica de programação, os desenvolvedores podem adaptar-se mais facilmente a diferentes linguagens de programação e ambientes de desenvolvimento, aproveitando ao máximo suas capacidades.

**Estruturas de Controle**

As estruturas de controle são responsáveis por determinar a ordem em que as instruções de um programa são executadas. As três principais estruturas de controle são:

**Sequência**: As instruções são executadas em ordem sequencial, uma após a outra.

**Seleção**: Permite que o programa tome decisões com base em condições. Exemplos incluem estruturas condicionais como "if", "else if" e "else".

**Repetição**: Permite que um conjunto de instruções seja repetido várias vezes, enquanto uma condição específica for verdadeira.

Exemplos incluem loops como "for", "while" e "do-while".

**Exemplos de Algoritmos com Estruturas de Controle**

Algoritmo de soma dos números pares de 1 a 10 (usando repetição). Algoritmo para verificar se um número é positivo, negativo ou zero (usando seleção). Algoritmo para calcular a média de uma lista de números (usando sequência e repetição).

**Tipos de Dados Básicos**

**Inteiros**: Representam números inteiros, como -3, 0, 42.

**Ponto flutuante**: Representam números decimais, como 3.14, -0.5, 2.718.

**Booleanos**: Representam valores de verdadeiro (true) ou falso (false).

**Caracteres**: Representam caracteres individuais, como 'a', 'B', '?'.

**Exemplos de Declaração e Utilização de Variáveis**

# Declaração e utilização de variáveis em Python

idade = 25

nome ="João"

salario = 1500.50

ativo = True

**Conceito de Função**

Uma função é um bloco de código que executa uma tarefa específica e pode ser chamado de qualquer lugar do programa. Elas ajudam a organizar o código, promovem a reutilização e facilitam a manutenção.

**Importância da Modularização**

A modularização é importante porque divide um programa em partes menores e mais gerenciáveis, chamadas de módulos ou funções. Isso torna o código mais fácil de entender, manter e depurar. Além disso, promove a reutilização do código, pois os módulos podem ser usados em diferentes partes do programa ou até mesmo em outros programas.

**Exemplos de Funções Simples:**

# Função para calcular a soma de dois números

def soma(a, b):

return a + b

# Função para verificar se um número é par

def par(numero):

return numero % 2 == 0