**SESC ESCOLA**

DANIEL RODRIGUES MARQUES BRANDÃO

**ALGORITMOS E LÓGICA DE PROGRAMAÇÃO**

Trabalho sobre algoritmos

Cuiabá

2024

DANIEL RODRIGUES MARQUES BRANDÃO

**ALGORITMOS E LÓGICA DE PROGRAMAÇÃO**

Trabalho sobre algoritmos

[Trabalho sobre algoritmos e lógica de programação] apresentado ao [CURSO DE PROGRAMAÇÃO DE JOGOS DIGITAIS]

Orientador: Wanderson

Cuiabá

2024

## 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 o 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

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

Algoritmos são sequências lógicas e finitas de instruções ou regras bem definidas que têm como objetivo final realizar uma tarefa ou resolver um problema específico. As instruções são seguidas de forma precisa, passo a passo, com o objetivo de atingir o objetivo desejado. Em termos mais gerais, um algoritmo é um conjunto de ações que orientam a solução de um problema. A relevância dos algoritmos na programação é crucial. Aqui estão algumas das razões:

Resolução de problemas: Algoritmos são a base para resolver problemas de forma estruturada e eficiente. Eles ajudam os programadores a dividir tarefas complexas em etapas mais fáceis e gerenciáveis.

Eficiência: Um bom algoritmo é concebido para ser eficiente em termos de tempo e recursos. Ele tem como objetivo finalizar uma tarefa da melhor forma possível e rápida. É possível reutilizá-los em diversas partes do programa ou em diferentes programas, o que reduz o tempo e o esforço do programador.

Observância e Colaboração: Algoritmos fornecem uma linguagem comum para que programadores expressem soluções para problemas. Isso torna mais fácil a compreensão do código por outros desenvolvedores, permitindo uma colaboração efetiva em projetos de software.

Automação: Regulamentos são cruciais para a automação de tarefas repetitivas. Eles permitem que máquinas realizem tarefas de forma consistente e segura.

A Base para o Desenvolvimento de Software: Todos os programas de computador são criados com base em algoritmos. Os softwares são a base por trás da criação de qualquer tipo de software.

Tomada de decisões: Algoritmos são bastante usados em sistemas de tomada de decisões, como algoritmos de inteligência artificial e aprendizado de máquina.

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

A lógica de programação é o conjunto de regras e princípios que orienta a elaboração de algoritmos de forma clara, coerente e eficiente. Ela serve como um guia para organizar pensamentos e estruturar a resolução de problemas de maneira lógica, facilitando a implementação prática por meio de linguagens de programação. A conexão entre lógica de programação e algoritmos é intrínseca, já que:

Organização Estruturada: A lógica de programação ajuda a organizar a solução de um problema em passos lógicos e estruturados. Essa organização é fundamental para a criação de algoritmos eficientes.

Tomada de decisões: A capacidade de tomar decisões com base em circunstâncias específicas é essencial para a lógica de programação. Os algoritmos frequentemente incluem estruturas condicionais (como instruções "se" e "senão") para lidar com diferentes casos.

Repetição (Laços de Controle): A lógica de programação trata da necessidade de repetir a execução de determinadas instruções. Os algoritmos incorporam estruturas de repetição (como loops) para lidar com tarefas que precisam ser realizadas várias vezes.

Abstração: Ambas buscam a habilidade de abstração, que consiste em simplificar um problema complexo, dividindo-o em partes menores e mais gerenciáveis. A lógica de programação auxilia na identificação dessas partes, enquanto os algoritmos detalham a solução de cada uma delas.

Entrada e saída de dados: A manipulação dos dados de entrada e saída é considerada tanto na lógica de programação quanto nos algoritmos. A lógica de programação fornece diretrizes para compreender e processar esses dados, enquanto os algoritmos orientam como executar essas tarefas de forma precisa.

Encapsulamento e Modularização: Ambas incentivam a prática de dividir um problema em partes menores, o que torna mais fácil a compreensão, a manutenção e a reutilização do código. A lógica de programação orienta essa prática, enquanto os algoritmos detalham como esses módulos funcionam juntos.

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

As estruturas de controle são fundamentais na programação, pois permitem que os desenvolvedores controlem o fluxo de execução do programa. As três principais estruturas de controle são sequência, seleção (ou condicional) e repetição (ou laço). Vamos explorar cada uma delas: Sequência: Definição: A estrutura de sequência refere-se à execução linear de instruções, onde uma instrução é executada após a outra, na ordem em que aparecem no código.

Seleção (ou Condicional): Definição: A estrutura de seleção permite que o programa tome decisões com base em condições específicas. Se uma condição for verdadeira, uma instrução ou bloco de instruções é executado; caso contrário, outro bloco de instruções pode ser executado.

Repetição (ou Laço): Definição: A estrutura de repetição permite que um bloco de instruções seja executado repetidamente enquanto uma condição específica for verdadeira. Isso é útil para automatizar tarefas repetitivas.

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

Exemplo 1: Sequência Objetivo: Calcular a média de dois números.

Exemplo 2: Seleção (Condicional) Objetivo: Verificar se um número é positivo, negativo ou zero.

Exemplo 3: Repetição (Laço) Objetivo: Exibir os números de 1 a 5.

Exemplo 4: Combinação de Estruturas Objetivo: Calcular o fatorial de um número.

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

Tipos de dados básicos são elementos fundamentais em programação que determinam a natureza dos valores que uma variável pode armazenar. Aqui estão algumas definições de tipos de dados básicos comuns:

Inteiro (int): Definição: Representa números inteiros, positivos ou negativos, sem parte fracionária. Exemplo: 1, -5, 1000.

Ponto Flutuante (float ou double): Definição: Representa números reais que podem ter uma parte fracionária. Pode armazenar valores decimais. Exemplo: 3.14, -0.5, 2.0.

Booleano (bool): Definição: Representa valores lógicos, verdadeiro (true) ou falso (false). Exemplo: true, false.

**EXEMPLOS DE DECLARAÇÃO E UTILIZAÇÃO DE VARIÁVEIS EM ALGORITMOS**

Exemplo 1: Variável Inteira Objetivo: Calcular a soma de dois números inteiros.

Exemplo 2: Variável Ponto Flutuante Objetivo: Calcular a média de dois números de ponto flutuante.

Exemplo 3: Variável Booleana Objetivo: Verificar se um número é positivo.

Exemplo 4: Variável String Objetivo: Concatenar duas strings.

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

O conceito de função na programação refere-se a um bloco de código autônomo, projetado para realizar uma tarefa específica, que pode ser chamado ou invocado a partir de outros locais no programa. Funções ajudam a organizar o código, promovem a reutilização e modularidade, facilitando a manutenção e compreensão do software. Aqui estão alguns pontos chave relacionados ao conceito de função na programação:

Definição de Função: Uma função é um conjunto de instruções agrupadas para realizar uma tarefa específica. Pode aceitar entrada (parâmetros), processá-la e retornar um resultado.

Declaração de Função: As funções geralmente são declaradas com um nome, lista de parâmetros e um bloco de código.

Chamada de Função: Para usar uma função, é necessário chamá-la fornecendo os argumentos necessários.

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

A modularização na escrita de algoritmos é de grande importância, pois oferece uma abordagem estruturada para o desenvolvimento de software, dividindo o código em módulos ou funções independentes. Essa prática traz uma série de benefícios significativos para o processo de programação e para o próprio software resultante.

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

Exemplo 1: Função para Calcular a Soma de Dois Números

Exemplo 2: Função para Verificar se um Número é Par