**SESC ESCOLA/JOGOS DIGITAIS**

JOÃO GABRIEL NEVES L. DE OLIVEIRA

**ALGORITMOS E LÓGICA DE PROGRAMAÇÃO**

Cuiabá

2024

JOÃO GABRIEL NEVES L. DE OLIVEIRA

**ALGORITMOS E LÓGICA DE PROGRAMAÇÃO**

Trabalho apresentado ao

Curso de Jogos Digitais

como requisito para obtenção de nota.

Orientador: Wanderson Timóteo

Cuiabá

2024

**SUMÁRIO**

**1 ALGORITMO.......................................................................................................4**

**1.2 DEFINIÇÃO DE ALGORITMOS E SUA IMPORTÂNCIA NA PROGRAMAÇÃO.....................................................................................4**

**1.3 LÓGICA DE PROGRAMAÇÃO E SUA RELAÇÃO COM ALGORITMOS.....................................................................................................4**

**2 ESTRUTURAS DE CONTROLE......................................................................4**

**2.2 PESQUISA SOBRE ESTRUTURAS DE CONTROLE COMO SEQUÊNCIA, SELEÇÃO E REPETIÇÃO...........................................................4**

**2.3 EXEMPLOS DE ALGORITMOS QUE UTILIZAM ESTRUT-**

**URAS DE CONTROLE.............................................................................5**

**3 TIPOS DE DADOS E VARIÁVEIS...................................................................5**

**3.2 DEFINIÇÃO DE TIPOS DE DADOS BÁSICOS, COMO IN-**

**TEIROS, PONTO FLUTUANTE, BOOLEANOS, ETC........................................5**

**3.3 EXEMPLOS DE DECLARAÇÃO E UTILIZAÇÃO DE VAR-**

**IÁVEIS EM ALGORITMOS.......................................................................7**

**4 FUNÇÕES E MODULARIZAÇÃO....................................................................7**

**4.2 PESQUISA SOBRE O CONCEITO DE FUNÇÃO NA PROGRAMAÇÃO................................................................................................7**

**4.3 A IMPORTÂNCIA DA MODULARIZAÇÃO NA ESCRITA**

**DE ALGORITMOS...............................................................................................8**

**4.4 EXEMPLOS DE FUNÇÕES SIMPLES E SUA UTILIZAÇÃO EM ALGORITMOS MAIS COMPLEXOS...................................................................8**

**ALGORITMOS**

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

Algoritmos são conjuntos de instruções sequenciais bem definidas e organizadas para resolver um problema ou executar uma tarefa específica.

Os algoritmos desempenham um papel fundamental na programação, fornecendo estrutura, eficiência e soluções para uma ampla variedade de problemas, desde os mais simples até os mais complexos. Dominar os algoritmos é essencial para se tornar um programador eficaz e bem-sucedido.

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

A lógica de programação é a capacidade de pensar de forma estruturada e sequencial para resolver problemas de forma eficiente utilizando um computador. Envolve a habilidade de desenvolver um raciocínio lógico para criar algoritmos.

a lógica de programação e os algoritmos estão intrinsecamente relacionados, pois ambos envolvem a aplicação de raciocínio lógico e estruturado para resolver problemas computacionais. Dominar a lógica de programação é essencial para criar algoritmos eficientes e desenvolver programas de alta qualidade.

**ESTRUTURAS DE CONTROLE**

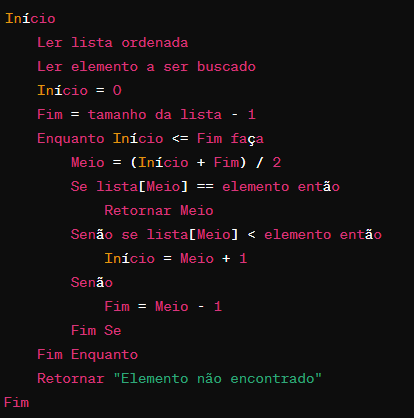
**PESQUISA SOBRE ESTRUTURAS DE CONTROLE COMO SEQUÊNCIA, SELEÇÃO E REPETIÇÃO**

As estruturas de controle são elementos fundamentais na programação, permitindo que os desenvolvedores controlem o fluxo de execução de um programa. Três das estruturas de controle mais comuns são: sequência, seleção e repetição.

O entendimento das estruturas de controle é essencial para escrever programas funcionais e eficientes. Elas permitem que os programadores controlem o comportamento do programa de acordo com as condições e requisitos específicos.

As estruturas de controle de sequência, seleção e repetição são elementos fundamentais na programação. Elas permitem que os desenvolvedores controlem o fluxo de execução de um programa e tomem decisões com base em condições específicas. Dominar essas estruturas é essencial para se tornar um programador competente e eficaz.

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

****

**TIPOS DE DADOS E VARIÁVEIS**

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

Os tipos de dados básicos são fundamentais na programação, pois fornecem uma maneira de representar diferentes tipos de informações e operar sobre elas. Aqui estão algumas definições dos tipos de dados básicos mais comuns:

Os inteiros são números inteiros sem parte fracionária. Eles podem ser positivos, negativos ou zero. Exemplos de inteiros incluem -3, 0, 42.

Os números de ponto flutuante representam números reais com uma parte fracionária. Eles podem incluir números decimais. Exemplos de ponto flutuante incluem 3.14, -0.001, 2.71828. O tipo "float" é usado para números de ponto flutuante de precisão simples, enquanto "double" é usado para números de ponto flutuante de precisão dupla.

Os booleanos representam valores lógicos que podem ser verdadeiros ou falsos. Eles são usados em expressões condicionais e de controle de fluxo. Exemplos de booleanos incluem true (verdadeiro) e false (falso).

Os caracteres representam caracteres individuais, como letras, números e símbolos. Cada caractere é armazenado como um código numérico de acordo com um conjunto de caracteres específico, como ASCII ou Unicode. Exemplos de caracteres incluem 'A', 'b', '7', '$'.

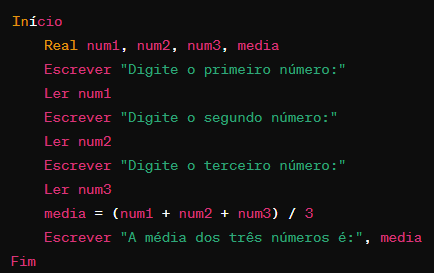
As strings são sequências de caracteres. Elas são usadas para representar texto e são amplamente utilizadas em programas para manipulação de dados de texto. Exemplos de strings incluem "Olá, mundo!", "12345", "foo bar".

Os ponteiros são variáveis especiais que armazenam endereços de memória de outras variáveis. Eles são usados para manipulação avançada de memória e para criar estruturas de dados mais complexas.

Além dos tipos de dados básicos mencionados acima, muitas linguagens de programação oferecem outros tipos de dados, como tipos de dados compostos (arrays, estruturas, classes), tipos de dados enumerados (enums), e tipos de dados definidos pelo usuário.

Esses tipos de dados básicos fornecem os blocos de construção essenciais para a criação de programas de computador, permitindo que os programadores representem e manipulem uma ampla variedade de informações de forma eficaz.

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

****

**FUNÇÕES E MODULARIZAÇÃO**

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

Em programação, uma função é um bloco de código que realiza uma tarefa específica. Ela pode receber entrada de dados (parâmetros), processá-los e retornar um resultado. As funções são usadas para organizar o código, torná-lo mais modular e reutilizável.

As funções geralmente têm uma estrutura composta por um cabeçalho e um corpo:

Cabeçalho: Inclui o nome da função, os parâmetros (se houver) e o tipo de dado que a função retorna (se aplicável).

Corpo: Contém as instruções que definem o comportamento da função.

As funções são fundamentais na programação, pois ajudam a organizar o código, promovem a reutilização de código e facilitam a manutenção e a depuração do programa. Elas permitem dividir um problema complexo em partes menores e mais gerenciáveis.

As funções são blocos de código essenciais na programação que permitem organizar e reutilizar código de forma eficaz, promovendo a modularidade e a manutenibilidade dos programas.

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

A modularização refere-se à prática de dividir um programa em módulos menores e independentes. Isso ajuda a organizar o código de forma mais lógica e compreensível, facilitando a manutenção e o entendimento do programa como um todo.

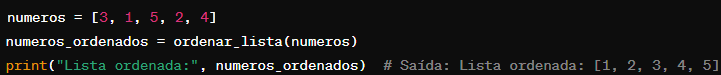
Ao dividir um algoritmo em módulos, cada parte do código se torna mais fácil de entender e manter. Os bugs podem ser localizados mais rapidamente e as alterações podem ser feitas de forma mais segura, sem afetar outras partes do programa.

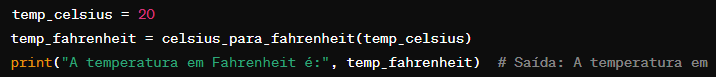
Programas modulares são mais fáceis de testar, pois cada módulo pode ser testado separadamente. Isso permite uma cobertura de teste mais completa e ajuda a garantir a qualidade e a confiabilidade do software.

A modularização promove a abstração e o encapsulamento, princípios importantes na programação orientada a objetos. Cada módulo pode representar uma abstração de uma funcionalidade específica e encapsular sua implementação interna.

Programas modulares tendem a ser mais legíveis e compreensíveis, especialmente para programadores que não estão familiarizados com o código. Os módulos bem nomeados e organizados facilitam o entendimento da lógica e da estrutura do programa.

Em resumo, a modularização é uma prática essencial na escrita de algoritmos, pois melhora a organização, a manutenção, a reutilização e a colaboração do código, ao mesmo tempo que promove a escalabilidade e a testabilidade do software.

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



Parte superior do formulário