



MIGUEL ROA AMARAL

**ALGORITMO E LÓGICA DE PROGROMAÇÃO**

E SUAS UTILIDADES

Cuiabá

2024

MIGUEL ROA AMARAL

**ALGORITMOS E LÓGICAS DE PROGRAMAÇÃO**

Trabalho apresentado no Sesc Escola, curso Jogos Digitais, Senac.

Orientador: Professor Wanderson

CUIABA

2024

**SUMÁRIO**

1. **ALGORITMO**
   1. Definição de algoritmos e sua importância na programação
   2. Lógica de programação e sua relação com algoritmos
2. **ESTRUTURAS DE CONTROLE**
   1. Pesquisa sobre estruturas de controle como sequência, seleção e repetição
   2. Exemplos de algoritmos que utilizam estruturas de controle
3. **TIPOS DE DADOS E VARIÁVEIS**
   1. Definição de tipos de dados básicos como inteiros, ponto flutuante, booleanos, etc.
   2. Exemplos de declaração e utilização de variáveis em algoritmos
4. **FUNÇÕES E MODULARIZAÇÕES**
   1. Pesquisa sobre o conceito de função na programação
   2. A importância da modularização na escrita de algoritmos
   3. Exemplos de funções simples e sua utilização em algoritmos mais complexos
5. **Algoritmo**

Um algoritmo é uma sequência de instruções que define um processo para resolver um problema. Algoritmos são usados em programação para automatizar tarefas e para construir programas que sejam capazes de resolver problemas complexos.

Também podendo ser separado em categorias de complexidade, o mais básico estando presente no processo de ligar o computador: O mesmo tem a função de testar e procurar componentes do sistema operacional.

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

A importância dos algoritmos na programação é que eles permitem que os programadores criem programas que são eficientes e eficazes. Os algoritmos também podem ser usados para resolver problemas que não podem ser resolvidos manualmente.

* 1. **Lógica de programação e sua relação com algoritmos**

A lógica de programação é um conjunto de regras que definem como os programas são escritos. Ela é baseada na lógica matemática e na teoria da computação, e é usada para descrever o comportamento de um programa.

Algoritmos são essenciais para o desenvolvimento de programas eficientes e eficazes. Eles permitem que os programadores criem programas que podem ser executados de forma rápida e precisa, e que podem resolver problemas complexos.

1. **Estruturas de Controle**

As estruturas de controle são instruções que controlam o fluxo de um programa. Elas permitem que o programador execute instruções em uma ordem específica, ou que execute instruções repetidamente até que uma condição seja satisfeita.

* 1. **Estruturas de controle de sequência, seleção e repetição**

Estrutura de Sequência: é uma estrutura de controle que executa as instruções em uma ordem específica. As instruções são executadas uma após a outra, até que todas sejam executadas. A estrutura de sequência é a estrutura de controle mais básica e é usada para executar uma série de instruções em uma ordem específica.

Estrutura de Seleção: permitem que o programador execute um conjunto de instruções se uma condição for verdadeira, ou outro conjunto de instruções se a condição for falsa. Existem três tipos de estrutura de seleção, sendo elas a simples, múltipla e aninhada.

Estrutura de Repetição: permitem que o programador execute um conjunto de instruções repetidamente até que uma condição seja satisfeita.

* 1. **Exemplos de algoritmos que utilizam estruturas de controle**

Algoritmo de cálculo de Fibonacci: é um algoritmo que calcula o número de Fibonacci. Ele pode ser implementado usando uma estrutura de repetição.

Arduino: utiliza a linguagem C++, onde seus códigos são postos a uma sequência, desde a maior prioridade até a menor. “void loop’ seguindo a estrutura de repetição, enquanto “if” e “else” segue uma estrutura de sequência.

1. **Tipos de dados e variáveis**

Na lógica de programação os dados são representados por variáveis, que são elementos capazes de armazenar um valor. A forma como um dado é representado na memória do computador depende do seu tipo, que pode ser numérico, alfanumérico, lógico ou outro. Cada tipo de dado tem um conjunto de regras que determinam como ele pode ser utilizado. Por exemplo, números inteiros podem ser somados e subtraídos, enquanto strings podem ser compatíveis.

* 1. **Definição de dados inteiros, ponto flutuante, booleanos, etc**

Dados Inteiros: são dados que podem ter apenas valores inteiros, positivos ou negativos. Eles são usados ​​para armazenar números que não têm uma parte decimal. são representados na memória do computador usando um número de bits, que é o número de dígitos que podem ser usados ​​para representar o número. Quanto maior o número de bits, maior o intervalo de valores que podem ser representados.

Ponto Flutuante: são números reais (0, 1, 2, 3, …) que podem ter uma parte inteira e uma parte decimal. Eles são representados na memória do computador usando um formato de ponto flutuante, que é um formato de armazenamento de números que permite representar números muito grandes e muito pequenos com precisão. São utilizados em aplicações matemáticas e científicas, mas a um problema, por serem números extensos, acabam sendo “arredondados” para um número real. Exemplo é o ponto flutuante são as raízes, como √2, √3.

Ponto Booleano: seguem a base de verdadeiro ou falso, são dados que representam valores lógicos. Na memória do computador, eles são representados em bit, 0 sendo falso e 1 sendo verdadeiros.

* 1. **Exemplos de declaração e utilização de variáveis em algoritmos**

// Declaração de uma variável inteira

int idade;

// Declaração de uma variável real

float altura;

// Declaração de uma variável booleana

bool é\_maior\_de\_idade;

// Atribuição de um valor a uma variável

idade = 20;

// Utilização de uma variável em uma expressão

altura = 1.70;

// Utilização de uma variável em uma condição

é\_maior\_de\_idade = idade >= 18;

1. **Funções e Modularização**

As funções são blocos de código que podem ser reutilizados em diferentes partes de um programa. Elas são definidas usando a palavra-chave def e podem ter parâmetros, que são variáveis que são passadas para a função quando ela é chamada.

Enquanto a modularização é o processo de dividir um programa em várias funções. Tornando o código mais fluido e otimizado, sendo possível executar várias funções em paralelo

* 1. **Exemplos de funções simples e sua utilização em algoritmos complexos**

Utilização de funções em algoritmos resulta na simplificação do código, o tornando modular. Por exemplo, um algoritmo que calcula o fatorial de um número poderia ser dividido em duas funções: uma função que calcula o fatorial de um número menor que 10 e outra função que calcula o fatorial de um número maior que 10.