

LÓGICA DE PROGRAMAÇÃO

Minicurso: Introdução Básica a Programação



DENISE MARTI

EBOOK

Sumário

SUMÁRIO

Introdução	3
Lógica de Programação	4
Pseudocódigo	6
Algoritmos	8
Variáveis	9
Estruturas de Controle.....	11
Funções.....	13
Considerações Finais	15
Informações sobre a autora	16

INTRODUÇÃO

Olá, tudo bem? Vamos embarcar em uma jornada para desvendar os mistérios da lógica de programação! Este curso rápido foi cuidadosamente elaborado especialmente para você, estudante iniciante em programação, que está dando os primeiros passos nesse mundo incrível da tecnologia.

Imagine a lógica de programação como um quebra-cabeça fascinante, onde cada peça representa uma instrução que o computador precisa seguir para executar uma tarefa. Neste curso, vamos desvendar os segredos por trás desse quebra-cabeça e aprender a montar nossos próprios algoritmos!

Ao final deste curso, vocês terão adquirido as habilidades necessárias para começar a conhecer as melhores linguagens de programação com confiança, e o melhor de tudo, sem nenhum conhecimento prévio sobre o assunto!

Vamos lá, embarque nessa jornada conosco e descubra o mundo da lógica de programação de uma forma simples e prática!

LÓGICA DE PROGRAMAÇÃO

A lógica de programação é a base para escrever programas de computador. Ela envolve pensar de forma estruturada, seguindo uma sequência lógica de instruções para resolver um problema. Aqui estão alguns conceitos importantes:

1 - Algoritmos: São sequências de passos bem definidos para realizar uma tarefa. Por exemplo, fazer um bolo tem um algoritmo: misturar os ingredientes, colocar no forno, esperar assar, etc.

2 - Variáveis: São "caixinhas" onde você guarda informações temporariamente, como números ou palavras. Por exemplo, você pode ter uma variável chamada "idade" que guarda a idade de uma pessoa.

3 - Estruturas de controle: São maneiras de controlar o fluxo de um programa. Os principais são:

a) Estruturas condicionais: Usadas para tomar decisões. Por exemplo, "se a idade for maior que 18, pode dirigir".

b) Estruturas de repetição: Usadas para repetir um bloco de código várias vezes. Por exemplo, "enquanto não chegar ao destino, continue andando".

4 - Funções: São blocos de código que realizam uma tarefa específica e podem ser reutilizados em diferentes

partes do programa. Por exemplo, uma função para somar dois números.

5 - Pseudocódigo: É uma forma de escrever algoritmos de forma mais próxima da linguagem humana, facilitando o entendimento antes de traduzir para uma linguagem de programação real.

PSEUDOCÓDIGO

O **pseudocódigo** é uma maneira simplificada de expressar um algoritmo. Ele é escrito em uma linguagem que é mais fácil para os humanos entenderem, em vez de uma linguagem de programação específica. O pseudocódigo não é executado em um computador, mas ajuda os programadores a entenderem e comunicar um algoritmo.

Aqui estão algumas características do pseudocódigo:

Linguagem Simples: O pseudocódigo é escrito em uma linguagem simples e compreensível para os humanos. Ele não segue a sintaxe rigorosa de uma linguagem de programação específica.

Foco no Lógico, não no Técnico: O pseudocódigo se concentra na lógica do algoritmo, não nos detalhes técnicos de uma linguagem de programação.

Independente de Linguagem: O pseudocódigo não está vinculado a nenhuma linguagem de programação específica. Isso significa que você pode implementar o algoritmo em qualquer linguagem de programação.

Aqui está um exemplo de como um algoritmo pode ser expresso em pseudocódigo:

Início

```
função calcularMedia(nota1, nota2)
```

```
    media = (nota1 + nota2) / 2
```

```
    retornar media
```

```
fim função
```

```
mediaFinal = calcularMedia(8, 9)
```

```
imprimir "A média final é: ", mediaFinal
```

Fim

snappify.com

ALGORITMOS

Um **algoritmo** é uma sequência de passos para resolver um problema ou realizar uma tarefa. Cada passo do algoritmo deve ser claro e executável. Aqui estão alguns conceitos importantes relacionados aos algoritmos:

Entrada e saída: Todo algoritmo requer entradas bem definidas e uma ou mais saídas. A entrada refere-se aos dados que são fornecidos para o algoritmo, enquanto a saída é o resultado do algoritmo.

Determinismo: Cada passo de um algoritmo deve ser determinístico, ou seja, deve ter um significado claro e deve levar a um único resultado.

Finitude: Um algoritmo deve sempre terminar após um número finito de passos.

Eficiência: Embora não seja um requisito estrito, geralmente queremos que nossos algoritmos sejam eficientes em termos de tempo e espaço.

```
Início
  Pegar os ingredientes
  Misturar os ingredientes
  Colocar a mistura no forno
  Esperar 30 minutos
  Verificar se o bolo está assado
  Se o bolo estiver assado, retirar do forno
  Se não, esperar mais 5 minutos e verificar novamente
Fim
```

snappify.com

VARIÁVEIS

Variáveis são um dos conceitos fundamentais em programação. Elas são usadas para armazenar informações que podem ser usadas em diferentes partes do seu programa. Aqui estão alguns pontos importantes sobre variáveis:

1 - Declaração de Variáveis: Antes de usar uma variável, você precisa declará-la. A declaração de uma variável reserva espaço na memória para armazenar valores. Ela envolve especificar o tipo de dados e o nome da variável.

2 - Atribuição de Variáveis: A atribuição de variáveis é o processo de dar um valor a uma variável já declarada.

3 - Tipos de Dados: Cada variável tem um tipo de dados. O tipo de dados determina o tipo de valor que a variável pode armazenar, como inteiros, flutuantes, caracteres, strings, booleanos, etc.

4 - Escopo de Variáveis: O escopo de uma variável refere-se à parte do programa onde a variável pode ser acessada. Variáveis podem ter escopo local (acessível apenas dentro de uma função específica) ou escopo global (acessível em todo o programa).

Aqui está um exemplo de como as variáveis podem ser usadas em pseudocódigo:




```
Início
  idade = 20
  imprimir "A idade é: ", idade
Fim

snappify.com
```

ESTRUTURAS DE CONTROLE

Elas são fundamentais na programação, pois permitem controlar o fluxo de execução do programa. Existem principalmente dois tipos de estruturas de controle: estruturas condicionais e estruturas de repetição.

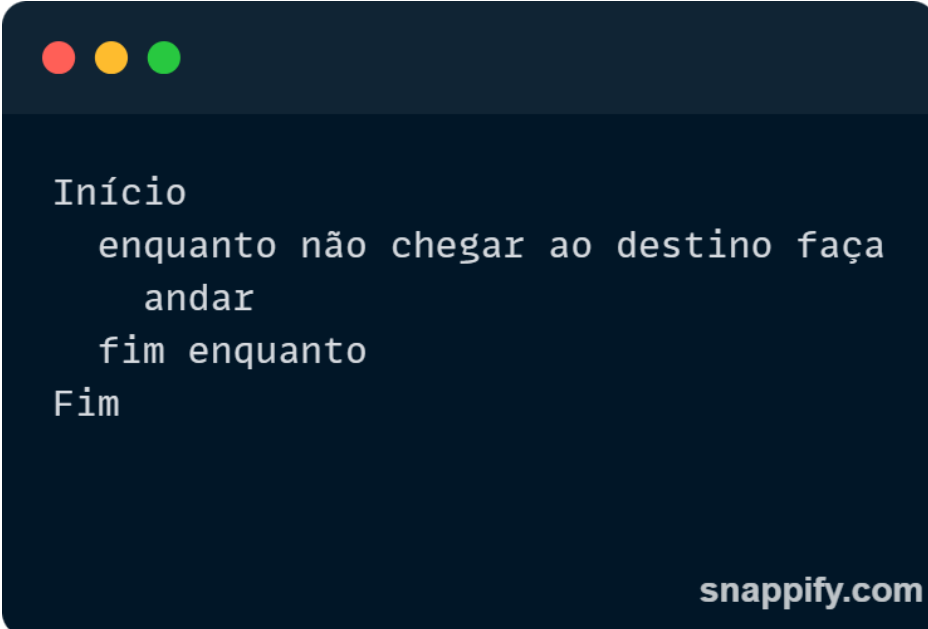
Estruturas condicionais: As estruturas condicionais, também conhecidas como instruções **if**, permitem que o programa tome decisões com base em condições. Por exemplo, em pseudocódigo, uma estrutura condicional



```
Início
  idade = 20
  se idade ≥ 18 então
    imprimir "Pode dirigir"
  senão
    imprimir "Não pode dirigir"
  fim se
Fim
```

Neste exemplo, o programa verifica se a idade é maior ou igual a 18. Se for, ele imprime “Você pode votar”. Se não for, ele imprime “Você não pode votar”.

Estruturas de repetição: As estruturas de repetição, também conhecidas como loops, permitem que o programa execute um bloco de código várias vezes. Existem vários tipos de loops, como **for**, **while** e **do-while**. Por exemplo, em pseudocódigo, um loop **for** pode ser assim:



```
Início
    enquanto não chegar ao destino faça
        andar
    fim enquanto
Fim
```

snappify.com

Neste exemplo, o programa imprime os números de 1 a 10. O loop **for** começa em 1 e termina em 10, incrementando **i** em 1 a cada iteração.

FUNÇÕES

Funções são blocos de código que realizam uma tarefa específica e podem ser reutilizados em diferentes partes do programa. Elas são fundamentais na programação, pois permitem escrever código de maneira mais modular e eficiente. Aqui estão alguns pontos importantes sobre funções:

Definição de Função: Uma função é definida por um nome, uma lista de parâmetros e um bloco de código. O bloco de código é o que a função executa quando é chamada.

Chamada de Função: Uma função é chamada pelo seu nome seguido por parênteses. Dentro dos parênteses, você pode passar argumentos para a função, que correspondem aos parâmetros definidos na função.

Retorno de Função: Uma função pode retornar um valor usando a palavra-chave **retornar**. Esse valor pode ser usado em outras partes do programa.

Escopo de Função: As variáveis definidas dentro de uma função têm escopo local, o que significa que só podem ser acessadas dentro da função. Variáveis definidas fora de todas as funções têm escopo global e podem ser acessadas em qualquer lugar do programa.

Início

```
// Definição de função
função somar(a, b)
    soma = a + b
    retornar soma
fim função

// Chamada de função
resultado = somar(3, 4)
imprimir "O resultado da soma é: ", resultado
Fim
```

snappify.com

Neste exemplo, a função **somar** recebe dois números, calcula a soma e retorna o resultado. A função é então chamada com os números 3 e 4, e o resultado é impresso.

CONSIDERAÇÕES FINAIS

Parabéns por concluir este minicurso em formato eBook! Agora possui os conhecimentos básicos e fundamentais da lógica de programação. Lembre-se, a lógica de programação é como uma superpoderosa ferramenta que permite que você transforme problemas complexos em soluções simples e eficazes. Com paciência, prática e um toque de criatividade, você pode dominar essa habilidade e se tornar um mestre da programação!

Então, não tenha medo de errar, pois cada erro é uma oportunidade de aprendizado. Continue praticando, explorando novos desafios e expandindo seus horizontes. Com o tempo, você verá que a lógica de programação se tornará uma segunda natureza para você.

Lembre-se sempre: a jornada da programação é como um jogo de aventura, cheio de surpresas e emoções. Divirta-se explorando novos conceitos, resolvendo problemas e criando soluções incríveis!

Agora, vá em frente e comece a transformar suas ideias em realidade. O mundo da programação está esperando por você!

INFORMAÇÕES SOBRE A AUTORA

Este EBOOK foi criado com a ajuda incrível da inteligência artificial e diagramado com carinho por Denise Marti. Sendo este, um Projeto do Bootcamp Nexa - Fundamentos de IA Generativa e Claude 3 chamado Natural ou Fake Natty? Como Vencer na Era das IAs Generativas!

O projeto original deste E-BOOK está disponível no meu perfil do GitHub.

<https://github.com/djeannie29/lab-natty-or-not/>