Plano de Estudo de 10 Dias: Lógica de Programação para Iniciantes Absolutos

Estrutura do Plano

Dia 1: Introdução a Variáveis e Tipos Primitivos Básicos

* Conceito de variável
* Convenções de nomenclatura
* Tipos primitivos: inteiros e decimais
* Exercícios práticos

Dia 2: Tipos Primitivos - Strings e Caracteres

* Tipo String
* Tipo Caractere
* Declaração e inicialização
* Exercícios práticos

Dia 3: Declaração, Tamanho em Memória e Casting Básico

* Tamanho dos tipos em memória
* Conversão entre tipos numéricos
* Casting implícito e explícito
* Exercícios práticos

Dia 4: Manipulação Básica de Strings

* Concatenação
* Comprimento de strings
* Acesso a caracteres
* Exercícios práticos

Dia 5: Operadores Aritméticos

* Adição, subtração, multiplicação, divisão
* Módulo (resto da divisão)
* Precedência de operadores
* Exercícios práticos

Dia 6: Operadores Relacionais

* Igualdade e diferença
* Maior, menor, maior ou igual, menor ou igual
* Comparações com diferentes tipos
* Exercícios práticos

Dia 7: Operadores Lógicos - AND

* Conceito de operador lógico AND
* Tabela verdade
* Aplicações práticas
* Exercícios práticos

Dia 8: Operadores Lógicos - OR

* Conceito de operador lógico OR
* Tabela verdade
* Combinação com AND
* Exercícios práticos

Dia 9: Operadores de Atribuição

* Atribuição simples
* Atribuição composta (+=, -=, \*=, /=, %=)
* Boas práticas
* Exercícios práticos

Dia 10: Desafio Final Integrado

* Revisão de todos os conceitos
* Desafio complexo combinando todos os temas
* Projeto prático final

Como Testar os Exercícios em Diferentes Linguagens

Python

# Exemplo de teste

variavel = 10

print(variavel)

JavaScript

// Exemplo de teste

let variavel = 10;

console.log(variavel);

Java

// Exemplo de teste

public class Main {

public static void main(String[] args) {

int variavel = 10;

System.out.println(variavel);

}

}

C

// Exemplo de teste

#include <stdio.h>

int main() {

int variavel = 10;

printf("%d\n", variavel);

return 0;

}

C++

// Exemplo de teste

#include <iostream>

using namespace std;

int main() {

int variavel = 10;

cout << variavel << endl;

return 0;

}

C#

// Exemplo de teste

using System;

class Program {

static void Main() {

int variavel = 10;

Console.WriteLine(variavel);

}

}

Dia 1: Introdução a Variáveis e Tipos Primitivos Básicos

Teoria

**Variáveis** são como caixas onde guardamos informações no computador. Cada caixa tem um nome (identificador) e armazena um valor. As **convenções de nomenclatura** são regras para nomear variáveis de forma legível, como camelCase (usado em Java, JavaScript, C#) ou snake\_case (usado em Python).

**Tipos Primitivos** definem a natureza do valor que uma variável pode armazenar. Os mais básicos são:

* **Inteiros:** Números inteiros, sem casas decimais (ex: 10, -5, 0).
* **Decimais (ou Ponto Flutuante):** Números com casas decimais (ex: 3.14, -0.5).

Exercícios

1. **Idade e Altura:** Declare uma variável para armazenar sua idade (inteiro) e outra para sua altura em metros (decimal). Imprima os dois valores na tela.
2. **Soma Simples:** Crie duas variáveis inteiras, numero1 e numero2, com valores de sua escolha. Crie uma terceira variável, soma, que armazene a soma das duas primeiras. Imprima o resultado.
3. **Área de um Retângulo:** Declare duas variáveis, base e altura, para um retângulo. Calcule a área (base \* altura) e armazene em uma variável area. Imprima o valor da área.

Dia 2: Tipos Primitivos - Strings e Caracteres

Teoria

* **String:** Uma sequência de caracteres, usada para armazenar texto. Strings são geralmente declaradas com aspas duplas ("texto") ou simples ('texto').
* **Caractere:** Um único símbolo, como uma letra, número ou pontuação. Em muitas linguagens, é declarado com aspas simples (ex: 'A').

Exercícios

1. **Nome Completo:** Declare uma variável do tipo string para armazenar seu nome completo e imprima-a.
2. **Inicial do Nome:** Declare uma variável do tipo caractere para armazenar a primeira letra do seu nome e imprima-a.
3. **Saudação:** Crie duas strings, uma com seu primeiro nome e outra com seu sobrenome. Concatene (junte) as duas com um espaço no meio para formar seu nome completo e imprima o resultado.

Dia 3: Declaração, Tamanho em Memória e Casting

Teoria

Cada tipo de dado ocupa um espaço diferente na **memória** do computador. Por exemplo, um inteiro pode ocupar 4 bytes, enquanto um decimal de precisão dupla pode ocupar 8 bytes. Isso é importante para otimização em sistemas complexos.

**Casting (ou conversão de tipos)** é o processo de transformar um valor de um tipo em outro.

* **Casting Implícito:** Ocorre automaticamente quando tipos compatíveis são misturados (ex: somar um inteiro com um decimal resulta em um decimal).
* **Casting Explícito:** É forçado pelo programador. Pode haver perda de informação (ex: converter um decimal para inteiro descarta a parte fracionária).

Exercícios

1. **De Decimal para Inteiro:** Declare uma variável decimal com o valor 9.99. Converta-a explicitamente para um tipo inteiro e imprima os dois valores. Note o que acontece com a parte fracionária.
2. **De Inteiro para Decimal:** Declare uma variável inteira com o valor 5. Converta-a para um tipo decimal e imprima os dois valores.
3. **Média de Notas:** Declare uma variável inteira nota1 e outra nota2. Calcule a média. Garanta que o resultado seja um número decimal, mesmo que a soma seja par. Imprima o resultado.

Dia 4: Manipulação Básica de Strings

Teoria

A manipulação de strings é fundamental em programação. As operações mais comuns são:

* **Concatenação:** Juntar duas ou more strings. O operador + é frequentemente usado para isso.
* **Comprimento:** Obter o número de caracteres em uma string.
* **Acesso a Caracteres:** Pegar um caractere específico de uma string usando sua posição (índice), que geralmente começa em 0.

Exercícios

1. **Tamanho da Palavra:** Crie uma string com uma palavra à sua escolha e imprima o número de caracteres que ela possui.
2. **Primeiro e Último Caractere:** Crie uma string e imprima o primeiro e o último caractere dela.
3. **Mensagem de Boas-Vindas:** Crie uma variável nome com um nome. Crie uma nova string que diga "Olá, [nome]! Seja bem-vindo(a)!", usando a variável nome. Imprima a mensagem final.

Dia 5: Operadores Aritméticos

Teoria

**Operadores Aritméticos** realizam cálculos matemáticos:

* + (Adição)
* - (Subtração)
* \* (Multiplicação)
* / (Divisão)
* % (Módulo): Retorna o resto de uma divisão inteira (ex: 10 % 3 resulta em 1).

A **precedência de operadores** define a ordem dos cálculos (multiplicação e divisão antes de adição e subtração).

Exercícios

1. **Conversor de Temperatura:** Crie uma variável celsius e converta seu valor para Fahrenheit. A fórmula é: F = C \* 1.8 + 32. Imprima o resultado.
2. **Calculadora de Média:** Declare três variáveis numéricas (n1, n2, n3). Calcule e imprima a média aritmética delas.
3. **Par ou Ímpar:** Crie uma variável inteira e use o operador de módulo (%) para verificar se o número é par ou ímpar. Um número é par se o resto da sua divisão por 2 for 0. Imprima uma mensagem indicando o resultado (ex: "O número X é par").

Dia 6: Operadores Relacionais

Teoria

**Operadores Relacionais** são usados para comparar dois valores. O resultado de uma comparação é sempre um valor booleano: verdadeiro (true) ou falso (false).

* == (Igual a): Verifica se dois valores são iguais.
* != (Diferente de): Verifica se dois valores são diferentes.
* > (Maior que)
* < (Menor que)
* >= (Maior ou igual a)
* <= (Menor ou igual a)

Exercícios

1. **Maioridade:** Crie uma variável idade e verifique se a pessoa é maior de idade (idade >= 18). Imprima true se for maior de idade e false caso contrário.
2. **Comparação de Números:** Declare duas variáveis numéricas, a e b. Compare-as e imprima mensagens indicando se a é maior, menor ou igual a b.
3. **Senha Correta:** Crie uma variável senhaArmazenada e outra senhaDigitada. Compare as duas para ver se são iguais. Imprima true se a senha estiver correta e false se estiver incorreta.

Dia 7: Operadores Lógicos - AND

Teoria

O operador lógico **AND** (representado como && em muitas linguagens, e and em Python) combina duas expressões booleanas. Ele retorna true somente se **ambas** as expressões forem verdadeiras.

**Tabela Verdade (AND):**

* true AND true -> true
* true AND false -> false
* false AND true -> false
* false AND false -> false

Exercícios

1. **Aprovação no Curso:** Para ser aprovado, um aluno precisa ter nota >= 7 E frequencia >= 75. Crie variáveis para nota e frequencia e verifique se o aluno foi aprovado. Imprima o resultado.
2. **Login Válido:** Um login é válido se o usuario é "admin" E a senha é "1234". Crie variáveis para usuario e senha e verifique se o login é válido.
3. **Intervalo Numérico:** Verifique se um número está no intervalo entre 50 e 100 (inclusive). Crie uma variável numero e use o operador AND para verificar se numero >= 50 E numero <= 100.

Dia 8: Operadores Lógicos - OR

Teoria

O operador lógico **OR** (representado como || em muitas linguagens, e or em Python) também combina duas expressões booleanas. Ele retorna true se **pelo menos uma** das expressões for verdadeira.

**Tabela Verdade (OR):**

* true OR true -> true
* true OR false -> true
* false OR true -> true
* false OR false -> false

Exercícios

1. **Entrada Gratuita:** A entrada em um evento é gratuita para crianças (idade < 12) OU idosos (idade > 65). Crie uma variável idade e verifique se a pessoa tem entrada gratuita.
2. **Fim de Semana:** Verifique se um dia da semana é sábado OU domingo. Crie uma variável diaDaSemana (string) e compare-a com "sabado" e "domingo".
3. **Compra de Produto:** Um cliente pode comprar um produto se tiver cartaoDeCredito válido OU saldoEmConta suficiente. Crie duas variáveis booleanas para representar essas condições e verifique se a compra é possível.

Dia 9: Operadores de Atribuição

Teoria

**Operadores de Atribuição** são usados para atribuir valores a variáveis. O mais simples é o =.

Os **operadores de atribuição composta** combinam uma operação aritmética com a atribuição, tornando o código mais conciso:

* += (Adição e atribuição): x += y é o mesmo que x = x + y
* -= (Subtração e atribuição): x -= y é o mesmo que x = x - y
* \*= (Multiplicação e atribuição): x \*= y é o mesmo que x = x \* y
* /= (Divisão e atribuição): x /= y é o mesmo que x = x / y
* %= (Módulo e atribuição): x %= y é o mesmo que x = x % y

Exercícios

1. **Contador Simples:** Inicie uma variável contador com 0. Use o operador += para incrementá-la em 1 cinco vezes, imprimindo o valor a cada incremento.
2. **Carrinho de Compras:** Inicie uma variável totalCompra com 100. Adicione 50, depois subtraia 20, depois multiplique por 2 e finalmente divida por 4. Use os operadores de atribuição composta para cada passo e imprima o resultado final.
3. **Pontuação de Jogo:** Crie uma variável pontuacao com 1000. Use os operadores de atribuição composta para simular a perda de 250 pontos, depois o ganho de 10% da pontuação atual. Imprima a pontuação final.

Dia 10: Desafio Final - Calculadora de Bônus de Funcionário

Teoria

Chegou a hora de aplicar tudo o que você aprendeu! O desafio de hoje combina o uso de variáveis de diferentes tipos, todos os operadores (aritméticos, relacionais, lógicos e de atribuição) e manipulação de strings. O objetivo é resolver um problema prático do mundo real, que exige a combinação lógica de todas essas ferramentas.

Desafio Prático

Você foi contratado para criar um programa simples que calcula o bônus de final de ano de um funcionário. As regras para o cálculo são as seguintes:

1. **Dados do Funcionário (Entrada):**
   * Nome do funcionário (string)
   * Salário base (decimal)
   * Nota de desempenho (decimal, de 0.0 a 1.0)
   * Anos de empresa (inteiro)
   * Possui uma ação disciplinar? (booleano: true ou false)
2. **Regras de Cálculo do Bônus:**
   * O **bônus base** é de 15% do salário base.
   * **Bônus de Desempenho:**
     + Se a nota de desempenho for maior que 0.8, o bônus base é **aumentado em 20%**.
     + Se a nota de desempenho for menor ou igual a 0.5, o bônus base é **reduzido em 10%**.
   * **Bônus de Antiguidade:**
     + Se o funcionário tiver mais de 5 anos de empresa **E** não tiver nenhuma ação disciplinar, ele recebe um bônus adicional de R$ 500,00.
   * **Condição Especial:**
     + Se o funcionário tiver uma nota de desempenho perfeita (1.0) **OU** tiver mais de 10 anos de empresa, ele recebe um bônus extra de R$ 200,00 (além de qualquer outro bônus).
   * **Penalidade:**
     + Se o funcionário tiver uma ação disciplinar, o bônus total é **reduzido em R$ 100,00**. O bônus nunca pode ser negativo (o valor mínimo é 0).
3. **Tarefa:**
   * Declare variáveis para cada um dos dados do funcionário e atribua valores a elas.
   * Calcule o bônus final seguindo todas as regras passo a passo.
   * Ao final, imprima uma mensagem formatada, como: "O funcionário [Nome] tem um bônus final de R$ [Valor do Bônus]."

**Exemplo de Teste:**

* **Entrada:**
  + Nome: "João Silva"
  + Salário Base: 5000.00
  + Nota de Desempenho: 0.9
  + Anos de Empresa: 6
  + Ação Disciplinar: false
* **Cálculo Esperado:**
  1. Bônus Base: 5000 \* 0.15 = 750
  2. Bônus de Desempenho: 750 \* 1.20 = 900 (nota > 0.8)
  3. Bônus de Antiguidade: 900 + 500 = 1400 (anos > 5 E sem ação)
  4. Condição Especial: Não se aplica (nota != 1.0 E anos <= 10)
  5. Penalidade: Não se aplica.
  6. **Bônus Final:** 1400
* **Saída:** "O funcionário João Silva tem um bônus final de R$ 1400.00."

Este desafio testará sua capacidade de organizar a lógica, usar parênteses para garantir a precedência correta das operações e combinar diferentes tipos de dados e operadores para chegar a um resultado final.

Dicas para Maximizar seu Aprendizado

Ao longo destes 10 dias, você estará construindo uma base sólida em lógica de programação. Para obter o máximo proveito deste plano de estudo, considere as seguintes recomendações:

**Pratique diariamente:** A consistência é fundamental para a aprendizagem de programação. Dedique pelo menos 1 a 2 horas por dia para estudar a teoria e resolver os exercícios propostos. Quanto mais você praticar, mais natural se tornará o processo de pensar logicamente e estruturar soluções.

**Teste em múltiplas linguagens:** Embora os conceitos sejam universais, experimentar diferentes linguagens de programação ajudará você a entender as nuances de cada uma e a desenvolver uma visão mais ampla. Comece com uma linguagem de sua preferência e, conforme ganhar confiança, teste os mesmos exercícios em outras linguagens. Python é recomendado para iniciantes devido à sua sintaxe simples, mas JavaScript, Java e C também são excelentes escolhas.

**Não tenha medo de errar:** Erros são parte essencial do aprendizado em programação. Quando seu código não funcionar como esperado, leia as mensagens de erro com atenção e tente entender o que deu errado. Use ferramentas de depuração ou imprima valores intermediários para rastrear problemas. Cada erro superado é uma lição valiosa.

**Documente seu progresso:** Mantenha um caderno ou arquivo digital onde você anota os conceitos aprendidos, dúvidas que surgiram e soluções para os exercícios. Isso facilitará a revisão e ajudará a consolidar o conhecimento.

**Busque recursos adicionais:** Além deste plano, explore tutoriais em vídeo, fóruns de programação e documentações oficiais das linguagens. Comunidades como Stack Overflow e Reddit podem ser úteis para tirar dúvidas específicas.

**Desafie-se além dos exercícios:** Após completar os exercícios de cada dia, tente criar variações ou problemas adicionais. Por exemplo, no Dia 5, você poderia criar um conversor de moedas ou uma calculadora de IMC. Essa prática autônoma acelerará seu desenvolvimento.

Tabela de Progressão de Conceitos

A tabela abaixo resume os conceitos abordados em cada dia do plano de estudo, permitindo que você acompanhe sua evolução e identifique áreas que podem precisar de revisão.

| Dia | Tema Principal | Conceitos-Chave | Nível de Dificuldade |
| --- | --- | --- | --- |
| 1 | Variáveis e Tipos Primitivos Básicos | Declaração, inteiros, decimais, convenções | Iniciante |
| 2 | Strings e Caracteres | Tipo string, tipo caractere, concatenação | Iniciante |
| 3 | Tamanho em Memória e Casting | Conversão de tipos, casting explícito/implícito | Iniciante |
| 4 | Manipulação de Strings | Comprimento, acesso a caracteres, concatenação | Iniciante |
| 5 | Operadores Aritméticos | Adição, subtração, multiplicação, divisão, módulo | Iniciante |
| 6 | Operadores Relacionais | Comparações, igualdade, maior/menor | Intermediário |
| 7 | Operadores Lógicos AND | Lógica booleana, tabela verdade, combinação de condições | Intermediário |
| 8 | Operadores Lógicos OR | Lógica booleana, tabela verdade, alternativas | Intermediário |
| 9 | Operadores de Atribuição | Atribuição simples, atribuição composta | Intermediário |
| 10 | Desafio Final Integrado | Combinação de todos os conceitos anteriores | Avançado |

Conclusão

Parabéns por se comprometer com este plano de estudo de 10 dias! Ao completar todos os exercícios e o desafio final, você terá desenvolvido uma compreensão sólida dos fundamentos da lógica de programação, incluindo tipos primitivos, operadores e manipulação de dados. Esses conceitos são a base sobre a qual toda a programação é construída, independentemente da linguagem ou tecnologia que você escolher explorar no futuro.

Lembre-se de que a jornada de aprendizado em programação é contínua. Após concluir este plano, você estará preparado para avançar para tópicos mais complexos, como estruturas de controle (condicionais e loops), funções, estruturas de dados e programação orientada a objetos. Continue praticando, explorando novos desafios e construindo projetos pessoais para consolidar e expandir seu conhecimento.

A programação é uma habilidade poderosa que abre portas para inúmeras oportunidades profissionais e criativas. Com dedicação e prática consistente, você estará no caminho certo para se tornar um programador competente e confiante. Boa sorte e bons estudos!

**Autor:** Manus AI  
**Data de Criação:** 07 de outubro de 2025