# Comandos Condicionais



# Introdução

Os comandos condicionais são estruturas fundamentais em programação que permitem que um programa tome decisões e execute diferentes blocos de código com base em condições específicas. Essas estruturas são essenciais para criar programas dinâmicos e interativos, capazes de responder a diferentes entradas e situações. Sem comandos condicionais, os programas seriam lineares e executariam sempre as mesmas instruções, independentemente das circunstâncias. Este material explora os principais tipos de comandos condicionais utilizados em programação, suas sintaxes, aplicações e boas práticas.

# Estrutura Condicional IF

A estrutura condicional IF é a mais básica e amplamente utilizada em programação. Ela permite que um bloco de código seja executado apenas se uma determinada condição for avaliada como verdadeira.

## Sintaxe básica do IF:

```
if (condição) {
    // bloco de código a ser executado se a condição for verdadeira
}
```

A condição dentro dos parênteses deve ser uma expressão que resulte em um valor booleano (verdadeiro ou falso). Se a condição for verdadeira, o bloco de código dentro das chaves será executado; caso contrário, o programa continuará a execução a partir da linha após o fechamento das chaves.

## Exemplo:

```
int idade = 18;
if (idade >= 18) {
    System.out.println("Você é maior de idade.");
}
```

Neste exemplo, a mensagem "Você é maior de idade" será exibida apenas se o valor da variável idade for maior ou igual a 18.
É importante notar que, em algumas linguagens de programação, as chaves podem ser omitidas se o bloco de código contiver apenas uma instrução. No entanto, por questões de clareza e para evitar erros, é recomendável sempre utilizar as chaves.

# Estrutura Condicional IF-ELSE

A estrutura IF-ELSE estende o conceito do IF simples, permitindo especificar um bloco de código alternativo a ser executado quando a condição for falsa.

## Sintaxe básica do IF-ELSE:

```
if (condição) {
    // bloco de código a ser executado se a condição for verdadeira
} else {
    // bloco de código a ser executado se a condição for falsa
}
```

Com esta estrutura, um dos dois blocos de código será sempre executado: se a condição for verdadeira, o primeiro bloco será executado; caso contrário, o segundo bloco (após o else) será executado.

## Exemplo:

```
int idade = 16;
if (idade >= 18) {
    System.out.println("Você é maior de idade.");
} else {
    System.out.println("Você é menor de idade.");
}
```

Neste exemplo, como a idade é 16 (menor que 18), a mensagem "Você é menor de idade" será exibida.
A estrutura IF-ELSE é particularmente útil quando há exatamente duas possibilidades mutuamente exclusivas e você deseja executar um código específico para cada uma delas.

# Estrutura Condicional IF-ELSE-IF

A estrutura IF-ELSE-IF (ou IF-ELIF-ELSE em algumas linguagens) permite verificar múltiplas condições em sequência, executando o bloco de código correspondente à primeira condição verdadeira encontrada.

## Sintaxe básica do IF-ELSE-IF:

```
if (condição1) {
    // bloco de código a ser executado se condição1 for verdadeira
} else if (condição2) {
    // bloco de código a ser executado se condição1 for falsa e condição2 for verdadeira
} else if (condição3) {
    // bloco de código a ser executado se condição1 e condição2 forem falsas e condição3 for verdadeira
} else {
    // bloco de código a ser executado se todas as condições anteriores forem falsas
}
```

As condições são verificadas na ordem em que aparecem. Assim que uma condição verdadeira é encontrada, o bloco de código correspondente é executado e as demais condições não são verificadas. O bloco else final é opcional e será executado apenas se todas as condições anteriores forem falsas.

## Exemplo:

```
int nota = 75;
if (nota >= 90) {
    System.out.println("Conceito A");
} else if (nota >= 80) {
```

```
    System.out.println("Conceito B");
} else if (nota >= 70) {
    System.out.println("Conceito C");
} else if (nota >= 60) {
    System.out.println("Conceito D");
} else {
    System.out.println("Conceito F");
}
```

Neste exemplo, como a nota é 75, a mensagem "Conceito C" será exibida.
A estrutura IF-ELSE-IF é útil quando há múltiplas possibilidades mutuamente exclusivas e você deseja executar um código específico para cada uma delas.

# Comandos Condicionais Aninhados

É possível aninhar comandos condicionais, ou seja, colocar um comando condicional dentro do bloco de código de outro comando condicional. Isso permite criar lógicas mais complexas e verificar condições dentro de condições.

## Exemplo:

```
int idade = 18;
boolean temCarteira = true;
if (idade >= 18) {
    if (temCarteira) {
        System.out.println("Você pode dirigir.");
    } else {
        System.out.println("Você tem idade para dirigir, mas precisa de uma carteira.");
    }
} else {
    System.out.println("Você não tem idade para dirigir.");
}
```

Neste exemplo, primeiro verificamos se a idade é maior ou igual a 18. Se for, verificamos se a pessoa tem carteira. Dependendo dessas condições, uma mensagem apropriada é exibida.
Embora os comandos condicionais aninhados sejam poderosos, eles podem tornar o código difícil de ler e manter se usados em excesso. Em muitos casos, é possível simplificar comandos aninhados usando operadores lógicos para combinar condições.

# Estrutura Condicional SWITCH-CASE

A estrutura SWITCH-CASE é uma alternativa ao IF-ELSE-IF quando se deseja comparar uma variável com múltiplos valores constantes. Ela oferece uma sintaxe mais clara e pode ser mais eficiente em termos de desempenho em alguns casos.

## Sintaxe básica do SWITCH-CASE:

```
switch (expressão) {
    case valor1:
        // bloco de código a ser executado se expressão == valor1
        break;
    case valor2:
        // bloco de código a ser executado se expressão == valor2
        break;
    case valor3:
        // bloco de código a ser executado se expressão == valor3
        break;
    default:
        // bloco de código a ser executado se expressão não corresponder a nenhum dos valores anteriores
}
```

A expressão é avaliada uma vez e seu valor é comparado com os valores de cada case. Se houver uma correspondência, o bloco de código associado a esse case é executado. A instrução break é usada para sair do switch após a execução do bloco de código. Se não houver break, a execução continuará nos cases subsequentes, independentemente de seus valores (comportamento conhecido como "fall-through"). O case default é opcional e será executado se nenhum dos cases anteriores corresponder ao valor da expressão.

## Exemplo:

```
int diaDaSemana = 3;
switch (diaDaSemana) {
    case 1:
        System.out.println("Domingo");
        break;
    case 2:
        System.out.println("Segunda-feira");
        break;
    case 3:
        System.out.println("Terça-feira");
        break;
case 5:
        System.out.println("Quinta-feira");
        break;
    case 6:
        System.out.println("Sexta-feira");
        break;
    case 7:
        System.out.println("Sábado");
        break;
    default:
        System.out.println("Dia inválido");
```

```
    case 4:
        System.out.println("Quarta-feira");
        break;
```

```
}
```

Neste exemplo, como diaDaSemana é 3, a mensagem "Terça-feira" será exibida.
A estrutura SWITCH-CASE é particularmente útil quando se deseja comparar uma variável com múltiplos valores constantes. No entanto, ela tem algumas limitações:

- Em muitas linguagens, os valores dos cases devem ser constantes (não podem ser variáveis ou expressões).
- Em linguagens tradicionais, os tipos de dados que podem ser usados na expressão do switch são limitados (geralmente inteiros, caracteres e, em algumas linguagens, strings).
- Não é possível usar operadores relacionais (como >, <, >=, <=) diretamente nos cases.

Algumas linguagens modernas, como C# e Java (a partir do Java 12), introduziram o "switch expression" ou "pattern matching", que amplia as capacidades do switch tradicional, permitindo comparações mais complexas.

# Operador Ternário

O operador ternário é uma forma concisa de escrever uma estrutura IF-ELSE simples. Ele é chamado de "ternário" porque envolve três operandos: uma condição, um valor a ser retornado se a condição for verdadeira e um valor a ser retornado se a condição for falsa.

## Sintaxe básica do operador ternário:

```
resultado = (condição) ? valorSeVerdadeiro : valorSeFalso;
```

## Exemplo:

```
int idade = 20;
String status = (idade >= 18) ? "Maior de idade" : "Menor de idade";
System.out.println(status);  // Saída: "Maior de idade"
```

Neste exemplo, como idade é 20 (maior que 18), a variável status receberá o valor "Maior de idade".
O operador ternário é útil para atribuições condicionais simples, mas pode tornar o código difícil de ler se usado em expressões complexas ou aninhado. É recomendável usá-lo apenas para casos simples e claros.

# Boas Práticas no Uso de Comandos Condicionais

Para garantir que seus comandos condicionais sejam claros, eficientes e livres de erros, considere as seguintes boas práticas:

- Use chaves mesmo para blocos de código de uma única linha. Isso torna o código mais claro e evita erros quando novas linhas são adicionadas.
- Evite condições complexas. Se uma condição se tornar muito complexa, considere dividi-la em partes menores ou usar variáveis booleanas intermediárias para melhorar a legibilidade.
- Prefira o SWITCH-CASE ao IF-ELSE-IF quando estiver comparando uma variável com múltiplos valores constantes.
- Evite o aninhamento excessivo de comandos condicionais. Se o aninhamento se tornar muito profundo, considere refatorar o código usando funções ou combinando condições.
- Tenha cuidado com o comportamento de "fall-through" no SWITCH-CASE. Sempre use break a menos que o comportamento de "fall-through" seja intencional e bem documentado.
- Use o operador ternário apenas para casos simples e claros. Para lógica condicional mais complexa, prefira o IF-ELSE.
- Considere a ordem das condições no IF-ELSE-IF. Coloque as condições mais prováveis ou mais simples primeiro para melhorar o desempenho e a legibilidade.
- Teste todas as ramificações condicionais para garantir que o comportamento seja o esperado em todos os casos.

# Conclusão

Os comandos condicionais são ferramentas essenciais na programação, permitindo que os programas tomem decisões e executem diferentes blocos de código com base em condições específicas. Neste material, exploramos os principais tipos de comandos condicionais: IF, IF-ELSE, IF-ELSE-IF, SWITCH-CASE e o operador ternário. Cada um desses comandos tem suas próprias características, vantagens e limitações, e a escolha entre eles depende do contexto específico e das necessidades do programa.

Ao dominar os comandos condicionais e seguir as boas práticas apresentadas, você estará equipado para criar programas mais dinâmicos, interativos e robustos. Lembre-se de que a clareza e a manutenibilidade do código são tão importantes quanto sua funcionalidade, por isso escolha a estrutura condicional mais apropriada para cada situação e organize seu código de forma lógica e compreensível.

# Referências


- 📖 Livros e Apostilas
  - CORMEN, T. H. Introduction to Algorithms. MIT Press.
  - GOODRICH, M. Data Structures and Algorithms in Python.
  - Tenenbaum, A. M. Estruturas de Dados e Algoritmos em C
  - P. Feofiloff. Algoritmos em Linguagem C. Campus-Elsevier, 1a. edição, 2009 H. M. Deitel, P. J. Deitel. C - Como Programar, 6a. edição, Pearson Education, 2011.
  - B. W. Kernighan, D. M. Ritchie. The C Programming Language, 2a. edição, Prentice-Hall, 1988 [Tradução: C - A Linguagem de Programação. Editora Campus, 1989].

    - J. L. Szwarcfiter, L. Markenzon. Estruturas de Dados e seus Algoritmos, 3a. edição, Editora LTC, 2010.
    - W. Celes, R. Cerqueira, J.L. Rangel. Introdução a Estruturas de Dados, 1a. edição, Editora Campus, 2004.
    - N. Ziviani. Projeto de Algoritmos com Implementações em Pascal e C, 3a. edição, Editora Cengage Learning, 2011.
    - T. Cormen, C. Leiserson, R. Rivest, C. Stein. Algoritmos - Teoria e Prática, 3a. edição, Editora Campus, 2012.
    - R. Sedgewick, K. Wayne. Algorithms, 4a. edição, Addison -Wesley, 2011.
    - A. Kelley, I. Pohl. A Book on C, 4a. edição, Addison Wesley, 1998.
- 🌐 Recursos Online
    - Utilizando comandos condicionais if-else e switch case - LinkedIn - https://pt.linkedin.com/pulse/utilizando-comandos-condicionais-if-else-e-switch-case-luciano-rocha
    - Instruções if e switch – selecionar um caminho de código para execução - Microsoft Learn - https://learn.microsoft.com/pt-br/dotnet/csharp/language-reference/statements/selection-statements
    - Estruturas de Decisão Condicional em Linguagem C - Bóson Treinamentos - https://www.bosontreinamentos.com.br/programacao-em-linguagem-c/estruturas-de-decisao-condicional-em-linguagem-c/
    - IF, SWITCH, FOR, WHILE e FOREACH em PHP - DevMedia - https://www.devmedia.com.br/if-switch-for-while-e-foreach-em-php/29527
    - Aula: Condicionais (if – else / switch) - UFOP - http://www3.decom.ufop.br/toffolo/media/uploads/2020-1/bcc201/07_condicionais_2.pdf
    - W3Schools - JavaScript Conditions
    - GeeksforGeeks - Decision Making in C/C++
- 📽️ Vídeos e Cursos
    - Curso em Vídeo - Estruturas Condicionais
    - Programação de Computadores - Estruturas Condicionais - UFOP
    - Khan Academy - Introdução à Programação

# Anexo - Resumo Estruturado/Mapa Mental Gerado por IA

🌟 Conceito Geral

- Permitem **tomada de decisões** em programas.
- Executam diferentes blocos de código com base em **condições lógicas**.
- Essenciais para criar **programas dinâmicos e interativos**.

---

🔑 Estruturas Condicionais

1. IF

- Mais simples e básica.
- Executa um bloco **somente se a condição for verdadeira**.

Sintaxe:
if (condição) { ... }

---

2. IF-ELSE

- Alternativa quando existem **duas possibilidades mutuamente exclusivas**.
- Um bloco executa se for verdadeiro, outro se for falso.

Sintaxe:
if (condição) { ... } else { ... }

---

3. IF-ELSE-IF

- Verifica **múltiplas condições em sequência**.
- Executa o bloco correspondente à **primeira condição verdadeira**.
- Possui bloco else opcional para "nenhuma condição satisfeita".

---

4. Condicionais Aninhados

- **IF dentro de IF** → permite lógica mais complexa.
- Útil para verificar **condições dependentes**.
- Risco: **código difícil de ler** → pode ser simplificado com operadores lógicos.

---

5. SWITCH-CASE

- Alternativa ao IF-ELSE-IF quando se compara uma variável a **valores constantes**.
- Mais **limpo e eficiente** em certos casos.
- Características:
    - Usa break para evitar *fall-through*.
    - Possui default para valores não previstos.
    - Limitado a valores constantes e tipos específicos (int, char, string em algumas linguagens).

---

6. Operador Ternário

- Forma **concisa do IF-ELSE simples**.

Sintaxe:
resultado = (condição) ? valorSeVerdadeiro : valorSeFalso;

- Bom para atribuições curtas, mas prejudica legibilidade em casos complexos.

---

✅ Boas Práticas

- Sempre usar **chaves**, mesmo em blocos de uma linha.
- Evitar **condições muito complexas** → preferir variáveis intermediárias.
- Usar **switch-case** em comparações com múltiplos valores fixos.
- Evitar **aninhamento excessivo**.
- Operador ternário → usar só em casos **simples e claros**.
- Ordenar condições no IF-ELSE-IF da **mais provável/simples para a mais complexa**.
- Testar **todas as ramificações** para garantir o funcionamento correto.

---

🎯 Conclusão

- Condicionais são fundamentais para dar **inteligência** aos programas.
- Tipos principais: **IF, IF-ELSE, IF-ELSE-IF, SWITCH-CASE, Operador Ternário**.
- Escolha depende do **contexto** e da **clareza desejada** no código.
- O bom uso garante **programas robustos, legíveis e fáceis de manter**.