Condicional

Variáveis "mágicas"

Em desenvolvimento chamamos de **variável mágica** aquela que armazena um valor ou expressão de forma que pode ser reaproveitada em vários pontos do código. Isso evita repetição e aumenta a legibilidade.

No .NET, o termo também pode aparecer para se referir a **constantes literais hardcoded** (“números mágicos”), que devem ser substituídos por variáveis com nome significativo.

Exemplo com expressão direta e usando variável:

```
// Sem a utilização de Variaveis magicas passamos os valores
// diretamente na expressão logica, e não temos noção do que representa seu
s valores
// nem oque esta validando.
if (17 < 18) // x < y?? -- Oque isso pode significar?
{
        // Temos noção do que se trata da validação na hora de apresentar o val
or.
    Console.WriteLine("Menor de Idade");
}

// usando variável mágica
// Com a utilização dessa pratica temos uma maior noção e legibilidade do có
digo,
// pois agora sabemos oque cada variavel representa.
int age = 17;
int maiorIdade = 18;
bool eMenorIdade = age < 18;

if (eMenorIdade)
{
```

```
    Console.WriteLine("Menor de Idade");
}
```

Condicionais no .NET (C#)

No desenvolvimento em C#, condicionais são usadas para **controlar o fluxo de execução**. Elas permitem que um bloco de código seja executado apenas quando uma condição lógica é satisfeita.

Essas condições sempre resultam em **true** ou **false**.

Um ponto importante é o uso de **variáveis mágicas** (aqui no bom sentido: variáveis intermediárias que armazenam expressões lógicas ou valores repetidos). Isso aumenta a legibilidade, facilita manutenção e evita “números mágicos” (constantes sem significado).

1. if-else

Forma mais básica de condicional. Usada quando precisamos tomar uma decisão simples binária.

Sintaxe:

```
if (condicao)
{
    // bloco executado se condicao for verdadeira
}
else
{
    // bloco executado se condicao for falsa
}
```

Com variável mágica:

```
int idade = 20;
bool podeVotar = idade >= 16; // variável mágica que descreve a regra
```

```
if (podeVotar)
    Console.WriteLine("Pode votar");
else
    Console.WriteLine("Não pode votar");
```

O uso da variável `podeVotar` evita espalhar a expressão `idade >= 16` em vários pontos.

2. if-elseif-else

Usado quando existem várias condições diferentes que podem ocorrer.

Sintaxe:

```
if (condicao1)
{
    // bloco 1
}
else if (condicao2)
{
    // bloco 2
}
else
{
    // bloco default
}
```

Com variável mágica:

```
int nota = 8;

bool excelente = nota >= 9;
bool bom = nota >= 7 && nota < 9;

if (excelente)
    Console.WriteLine("Excelente");
else if (bom)
    Console.WriteLine("Bom");
```

```
else
    Console.WriteLine("Reprovado");
```

Aqui, cada expressão lógica é armazenada em uma variável descritiva (`excelente`, `bom`), o que torna a leitura mais clara.

3. **switch-case**

Mais legível do que múltiplos `if-elseif` quando estamos comparando o **mesmo valor** contra vários cenários possíveis.

Sintaxe tradicional:

```
switch (valor)
{
    case opcao1:
        // bloco
        break;
    case opcao2:
        // bloco
        break;
    default:
        // bloco default
        break;
}
```

Exemplo com variável mágica:

```
string dia = "Segunda";

switch (dia)
{
    case "Segunda":
        Console.WriteLine("Início da semana");
        break;
    case "Sexta":
        Console.WriteLine("Fim de semana chegando");
```

```
        break;
    default:
        Console.WriteLine("Dia comum");
        break;
}
```

Switch expression (C# moderno):

```
string mensagem = dia switch
{
    "Segunda" => "Início da semana",
    "Sexta" => "Fim de semana chegando",
    _ => "Dia comum"
};
Console.WriteLine(mensagem);
```

Esse formato é mais compacto e normalmente utilizado para **atribuição de valores**.

4. Operador ternário (`?:`)

É um **atalho para if-else** que retorna um valor em apenas uma linha.

Útil quando precisamos **atribuir valores a uma variável** com base em uma condição.

Sintaxe:

```
variavel = condicao ? valor_se_verdadeiro : valor_se_falso;
```

Exemplo:

```
int idade = 20;
string categoria = idade >= 18 ? "Adulto" : "Menor";
```

Com variável mágica para clareza:

```
int idade = 20;
bool maiorDeIdade = idade >= 18; // variável mágica

string categoria = maiorDeIdade ? "Adulto" : "Menor";
Console.WriteLine(categoria);
```

Aqui a variável `maiorDeIdade` torna a expressão mais legível, principalmente quando a condição é mais complexa (exemplo: `idade >= 18 && possuiCNH`).

💡 Quando usar cada tipo

- **if-else:** para decisões binárias simples.
- **if-elseif-else:** para múltiplas faixas de valores ou condições distintas.
- **switch-case:** quando temos **um valor específico** sendo comparado a várias opções.
- **operador ternário:** quando a condição serve para **atribuir valores rapidamente**.

Lista de 100 exercícios para praticar condicionais (.NET)

1. Verifique se um número é positivo ou negativo.
2. Verifique se um número é par ou ímpar.
3. Compare dois números e mostre o maior.
4. Verifique se uma pessoa pode votar (idade >= 16).
5. Verifique se uma pessoa pode dirigir (idade >= 18).
6. Calcule a média de 2 notas e mostre "Aprovado" ou "Reprovado".
7. Calcule a média de 3 notas e classifique em aprovado, recuperação ou reprovado.
8. Verifique se uma senha digitada é igual a "1234".

9. Verifique se um ano é bissexto.
10. Classifique a idade em criança, adolescente, adulto ou idoso.
11. Mostre o dia da semana com switch-case.
12. Classifique uma letra como vogal ou consoante.
13. Receba um número e mostre se está entre 1 e 100.
14. Calcule desconto: se valor > 100 aplicar 10%.
15. Verifique se um número é múltiplo de 5.
16. Verifique se um número é múltiplo de 3 e de 5.
17. Calcule o IMC e classifique (baixo peso, normal, sobrepeso, obesidade).
18. Mostre a estação do ano de acordo com o mês.
19. Verifique se uma string contém "@" para validar e-mail simples.
20. Receba três números e mostre o maior.
21. Receba três números e mostre o menor.
22. Verifique se três lados podem formar um triângulo.
23. Classifique o triângulo como equilátero, isósceles ou escaleno.
24. Receba um caractere e verifique se é dígito.
25. Verifique se uma palavra é palíndromo.
26. Use operador ternário para verificar se um número é par.
27. Calcule bônus: salário < 2000 ganha 20%, caso contrário 10%.
28. Classifique notas em A, B, C, D ou F.
29. Verifique se a soma de dois números é maior que 100.
30. Receba dois números e verifique se um é múltiplo do outro.
31. Simule login: usuário e senha devem bater.
32. Classifique a hora: manhã, tarde ou noite.
33. Verifique se um caractere é maiúsculo ou minúsculo.
34. Valide CPF com quantidade de dígitos (11).

35. Receba o mês (1 a 12) e mostre a quantidade de dias.
36. Receba uma nota de 0 a 10 e valide se está dentro da faixa.
37. Jogo pedra-papel-tesoura usando switch-case.
38. Receba idade e verifique se pode entrar em um evento (+21).
39. Verifique se um número é primo.
40. Receba dois números e escolha a operação com switch (soma, subtração etc.).
41. Verifique se uma senha tem mais de 8 caracteres.
42. Mostre a categoria de um atleta por idade.
43. Use switch para classificar um número de 1 a 7 em dia da semana.
44. Receba um número decimal e classifique como inteiro ou não.
45. Receba nota e transforme em conceito (>=90 A, >=80 B...).
46. Verifique se um número é dentro do intervalo [10, 50].
47. Receba três notas e ignore a menor no cálculo da média.
48. Verifique se uma palavra começa com vogal.
49. Receba duas strings e verifique se são iguais ignorando maiúsculas/minúsculas.
50. Classifique idade em faixas etárias com switch (0–12, 13–17, etc.).
51. Verifique se uma senha contém número.
52. Verifique se uma senha contém caractere especial.
53. Receba salário e calcule faixa de imposto (simulação).
54. Verifique se um número é potência de 2.
55. Verifique se o caractere digitado é espaço.
56. Receba hora e mostre "Bom dia", "Boa tarde" ou "Boa noite".
57. Calcule nota final considerando peso em duas provas.
58. Receba altura e sexo, calcule peso ideal.

59. Verifique se um ponto (x,y) está no 1º, 2º, 3º ou 4º quadrante.
60. Receba idade e classifique como "iniciante", "intermediário", "avançado".
61. Verifique se o número digitado termina em 0.
62. Verifique se um número tem 3 dígitos.
63. Receba valor e verifique se cabe no cartão (limite).
64. Verifique se o mês informado é férias (dezembro, janeiro, julho).
65. Receba dois horários e mostre qual é maior.
66. Calcule aumento de salário: se < 1500 → 20%, senão 10%.
67. Receba nota e dê mensagem de acordo com faixa (if-elseif).
68. Receba temperatura em °C e classifique (frio, agradável, quente).
69. Receba número e verifique se é divisível por 2 e 7.
70. Receba idade e verifique se pode se aposentar (homem 65, mulher 62).
71. Receba data e verifique se é fim de semana.
72. Receba número e verifique se é maior que 0, menor que 0 ou zero.
73. Receba 4 notas e descarte a menor.
74. Use switch para operações matemáticas (+, -, *, /).
75. Receba número e mostre se é perfeito (soma dos divisores = número).
76. Verifique se um número é capicua (igual ao reverso).
77. Receba string e verifique se termina com ".com".
78. Receba idade e mostre se pode entrar em balada (+18).
79. Receba string e verifique se contém apenas números.
80. Receba salário e calcule plano de saúde (faixa etária).
81. Receba ano e verifique se é do século XX ou XXI.
82. Receba valor e classifique como caro ou barato com base em limite.
83. Receba três números e ordene-os (usando if).
84. Verifique se um número está entre dois valores.

35. Receba um número e classifique como baixo, médio ou alto.
36. Receba número de horas trabalhadas e calcule hora extra.
37. Receba dois números e mostre se são iguais.
38. Receba uma letra e verifique se é do alfabeto.
39. Receba temperatura em Fahrenheit e classifique.
90. Receba idade e mostre faixa escolar.
91. Receba valor e verifique se pode ser dividido em 3 parcelas iguais.
92. Receba número e verifique se é quadrado perfeito.
93. Receba idade e mostre se precisa de autorização dos pais.
94. Receba número e verifique se cabe em um byte.
95. Receba string e verifique se tem espaços no início ou fim.
96. Receba nota e mostre feedback automático.
97. Receba número e verifique se é divisor de 100.
98. Receba idade e verifique se pode participar de competição (>=14).
99. Receba número e verifique se está no intervalo aberto (10, 20).
00. Receba nota e mostre conceito com operador ternário.

Reversão e abstração de lógica em condicionais

Quando falamos em **reversão de lógica**, estamos tratando de inverter a expressão condicional para simplificar a leitura do código.

Já **abstração de lógica** significa extrair expressões complexas ou repetidas e armazenar em **variáveis mágicas** (variáveis booleanas com nomes descritivos), deixando o código mais claro e menos sujeito a erros.

---

1. **Reversão de lógica (early return / guard clause)**

Muitos códigos ficam confusos porque usam condicionais “aninhadas demais”. A técnica de **early return** reduz profundidade invertendo a lógica:

```
// Exemplo ruim (aninhamento)
if (usuario != null)
{
    if (usuario.Ativo)
    {
        Console.WriteLine("Acesso permitido");
    }
}

// Melhor: early return
if (usuario == null || !usuario.Ativo)
    return; // finaliza aqui

Console.WriteLine("Acesso permitido");
```

- Aqui a **reversão** elimina blocos aninhados.
- O retorno antecipado (`return`) deve ser usado quando **não faz sentido continuar a execução** após a falha.

2. **Abstração com variáveis mágicas**

Expressões repetidas ou complexas devem ser isoladas em variáveis booleanas. Isso gera nomes autoexplicativos.

```
int idade = 20;
bool maiorDeIdade = idade >= 18;   // variável mágica
bool podeBeber = maiorDeIdade;     // regra abstraída

if (podeBeber)
    Console.WriteLine("Pode consumir bebida alcoólica");
```

Se a regra mudar (ex.: limite legal passa para 21), basta alterar a variável mágica.

3. **Aplicando em cada tipo de condicional**

- **if-else:** use variáveis mágicas quando a condição tiver mais de uma comparação.

```
bool autorizado = usuario != null && usuario.Ativo && usuario.Idade >= 18;

if (autorizado)
    Console.WriteLine("Acesso permitido");
else
    Console.WriteLine("Acesso negado");
```

- **if-elseif-else:** crie flags descritivas para cada faixa.

```
bool crianca = idade < 12;
bool adolescente = idade >= 12 && idade < 18;

if (crianca)
    Console.WriteLine("Criança");
else if (adolescente)
    Console.WriteLine("Adolescente");
else
    Console.WriteLine("Adulto");
```

- **switch expression:** simplifique com variáveis quando há cálculos repetidos.

```
int idade = 20;
bool maior = idade >= 18;

string categoria = maior switch
{
    true  ⇒ "Adulto",
    false ⇒ "Menor"
};
```

- **Operador ternário:** ótimo para atribuição rápida, mas se a condição ficar longa, abstraia.

```
bool ativo = usuario?.Ativo ?? false;
string status = ativo ? "Ativo" : "Inativo";
```

---

4. **Quando finalizar forçadamente (`return`, `break`, `continue`, `throw`)**

- **Use retorno antecipado** quando o resto do código **não faz sentido** se a condição falhar.
- **Não use** se houver necessidade de avaliar múltiplas condições independentes.
- `break` é usado em loops ou `switch` para interromper a execução.
- `continue` pula para a próxima iteração em um loop.
- `throw` finaliza com exceção e deve ser usado apenas em cenários excepcionais, não em fluxo comum.

Exemplo:

```
if (usuario == null)
    throw new ArgumentNullException(nameof(usuario));

if (!usuario.Ativo)
    return; // não precisa seguir

Console.WriteLine("Processando usuário...");
```

O conceito de **flag**

Uma **flag** é uma variável (geralmente booleana) usada para **indicar um estado** ou **controlar o fluxo** de execução.

- Pode servir como "interruptor" em loops ou fluxos.

- Muitas vezes substitui condicionais complexas por uma variável de estado clara.

Exemplo com flag simples:

```
bool encontrado = false;

foreach (var item in lista)
{
    if (item == "X")
    {
        encontrado = true; // setamos a flag
        break;             // interrompemos o loop
    }
}

if (encontrado)
    Console.WriteLine("Item encontrado");
else
    Console.WriteLine("Item não existe");
```

Exemplo com flag de controle em processo:

```
bool precisaRecalcular = true;

if (precisaRecalcular)
{
    RecalcularEstoque();
    precisaRecalcular = false; // desarma a flag
}
```

Flags evitam reprocessamento desnecessário e tornam claro em que estado o sistema está.

Cenário: validar usuário antes de processar um pedido

❌ Código “ruim” (aninhado, difícil de manter)

```
if (usuario != null)
{
    if (usuario.Ativo)
    {
        if (usuario.Idade >= 18)
        {
            Console.WriteLine("Pedido processado.");
        }
        else
        {
            Console.WriteLine("Usuário menor de idade.");
        }
    }
    else
    {
        Console.WriteLine("Usuário inativo.");
    }
}
else
{
    Console.WriteLine("Usuário inválido.");
}
```

Problemas:

- Muitas camadas de `if` aninhados.
- A lógica principal ("Pedido processado") fica escondida no fundo.
- Difícil de alterar sem risco de erro.

✅ Código “bom” (early return + variáveis mágicas)

```
bool usuarioValido = usuario != null;
bool usuarioAtivo = usuario?.Ativo ?? false;
```

```
bool maiorDeIdade = usuario?.Idade >= 18;

if (!usuarioValido)
{
    Console.WriteLine("Usuário inválido.");
    return;
}

if (!usuarioAtivo)
{
    Console.WriteLine("Usuário inativo.");
    return;
}

if (!maiorDeIdade)
{
    Console.WriteLine("Usuário menor de idade.");
    return;
}

Console.WriteLine("Pedido processado.");
```

Vantagens:

- **Reversão de lógica**: cada caso inválido finaliza cedo (`return`), liberando o fluxo principal.
- **Variáveis mágicas** (`usuarioValido`, `usuarioAtivo`, `maiorDeIdade`) tornam as regras legíveis.
- O objetivo principal (“processar pedido”) aparece limpo, sem ruído.

📌 **Resumo da comparação**:

- ❌ **Código ruim**: aninhamento profundo → mais difícil de ler e manter.
- ✅ **Código bom**: early return + variáveis mágicas → mais direto, autoexplicativo, fácil de evoluir.

🔷 Comparações numéricas e lógicas

1. Verifique se um número é múltiplo de 3 e 5 ao mesmo tempo.
2. Cheque se um número está entre 10 e 50, **mas não é 30**.
3. Teste se dois números possuem o mesmo sinal (ambos positivos ou ambos negativos).
4. Determine se três números formam uma sequência crescente.
5. Verifique se um número é primo com base em divisões condicionais.
6. Verifique se o maior de dois números é par.
7. Cheque se um número está dentro de um intervalo fechado ou aberto (ex.: `[10, 20]` ou `(10,20)`).
8. Faça um programa que verifica se uma fração `a/b` é maior que outra `c/d`.
9. Crie uma condição que só aceita números maiores que 100 ou iguais a 42.
10. Valide se a soma de dois números é par e o produto é ímpar.

🔷 Strings e validações

1. Verifique se uma string começa com "A" e termina com "Z".
2. Teste se uma string contém apenas letras maiúsculas.
3. Verifique se uma senha contém ao menos 1 letra, 1 número e 1 caractere especial.
4. Cheque se duas strings são iguais ignorando maiúsculas/minúsculas.
5. Teste se o email informado possui `@` e um domínio válido (`.com`, `.org`).
6. Verifique se uma palavra é palíndromo.
7. Crie uma condicional que diferencie CPF de CNPJ (com base no tamanho da string).
8. Verifique se uma URL começa com `https://`.
9. Teste se o nome possui mais de 2 palavras.

10. Crie uma condicional para validar formato de placa veicular (padrão antigo e Mercosul).

🔷 Operador ternário e atribuições

1. Use ternário para decidir entre `"Maioridade"` ou `"Menoridade"`.
2. Atribua um valor padrão `0` se a entrada for `null`.
3. Defina a cor `"Verde"` se status for `"Ativo"`, senão `"Vermelho"`.
4. Crie um ternário aninhado que decide o nível de nota (A, B, C, D).
5. Aplique ternário para definir desconto: `10%` se cliente VIP, `5%` se comum.
6. Defina mensagem `"Aprovado"`, `"Recuperação"` ou `"Reprovado"`.
7. Retorne o menor de dois números usando apenas ternário.
8. Crie uma condicional ternária que retorna `"Login OK"` ou `"Acesso negado"`.
9. Aplique ternário para exibir `"Sim"` se booleano `true`, `"Não"` se `false`.
10. Atribua `"Desconhecido"` se a variável `nome` estiver vazia.

🔷 Switch-case (complexos)

1. Implemente um menu usando `switch` para CRUD.
2. Decida a estação do ano com base no mês informado.
3. Retorne o nome do dia da semana (1 = Domingo, 7 = Sábado).
4. Identifique se o mês informado tem 28, 30 ou 31 dias.
5. Use `switch` para calcular o imposto com base no estado (SP, RJ, MG).
6. Crie um conversor de unidade (km → mi, kg → lb).
7. Identifique categoria de veículo (Moto, Carro, Caminhão).
8. Use `switch` para traduzir status: `"P" = Pago, "A" = Atrasado, "C" = Cancelado`.
9. Crie `switch` com **fallthrough** (quando vários cases têm mesmo resultado).
10. Implemente um jogo simples de pedra-papel-tesoura com `switch`.

🔷 Flags e controle de fluxo

1. Crie um `bool flag` que indica se login foi bem-sucedido.
2. Use uma flag para interromper um laço ao encontrar um número negativo.
3. Use flag para verificar se todos os números de uma lista são positivos.
4. Faça um sistema que só continua se a flag de "termos aceitos" for true.
5. Implemente uma flag para alternar modo claro/escuro.
6. Use flag para marcar se um arquivo foi carregado corretamente.
7. Crie flag para validar se CPF foi digitado **e** validado.
8. Monte flag `isDirty` para saber se houve alterações em um formulário.
9. Use flag para parar leitura de lista ao encontrar valor nulo.
10. Crie flag `gameOver` em um joguinho de adivinhação.

🔷 Reversão de lógica

1. Reescreva `if (x == false)` como `if (!x)`.
2. Simplifique `if (x != true)` usando reversão.
3. Inverta uma condição `if (a > b)` para `if (!(a <= b))`.
4. Transforme `if-else` longo em `return` antecipado.
5. Refatore `if (condicao) { return true; } else { return false; }`.
6. Troque `if (!(idade < 18))` por `if (idade >= 18)`.
7. Converta um `switch` em vários `if-else`.
8. Reescreva um ternário complexo em `if-else`.
9. Simplifique condições com `||` e `&&` eliminando redundâncias.
10. Inverta lógica de loop para reduzir nesting.

🔷 Casos práticos (nível real)

1. Valide se o usuário tem idade suficiente e não está bloqueado.

2. Verifique se um carrinho tem produtos e se o estoque está disponível.
3. Condicional para aplicar frete grátis apenas acima de 200 reais.
4. Permitir login apenas se email confirmado e senha válida.
5. Exiba "Promoção ativa" apenas entre duas datas.
6. Verifique se dois horários se sobrepõem.
7. Controle de acesso: admin, gerente, usuário comum.
8. Identifique se uma pessoa pode votar (idade >= 16) e se é obrigatório.
9. Determine se um pagamento é válido: não vencido e não cancelado.
10. Desconto progressivo: até 5 itens = 5%, até 10 = 10%, acima de 10 = 20%.

🔷 Estruturas aninhadas

1. Condicional para validar login e, dentro, validar permissão.
2. Teste se um triângulo é equilátero, isósceles ou escaleno.
3. Determine se ano é bissexto (divisível por 4, mas não por 100, exceto por 400).
4. Verifique se um número é positivo, negativo ou zero.
5. Valide se um número pertence a Fibonacci.
6. Controle de notas: maior que 7 aprovado, entre 5-7 recuperação, abaixo reprovado.
7. Cheque se uma data é válida (dia compatível com mês).
8. Sistema de login: 3 tentativas antes de bloquear.
9. Sistema de caixa eletrônico: só saque múltiplos de 10.
10. Determine categoria de IMC (baixo, normal, sobrepeso, obeso).

🔷 Misturando conceitos

1. Combine `switch` com ternário.

2. Use flags dentro de `switch` para validar cenários.
3. Use condicional reversa para reduzir profundidade de nesting.
4. Transforme validações longas em métodos auxiliares com `return`.
5. Crie método que retorna "Aprovado" apenas se média >= 7 **e** frequência >= 75%.
6. Implemente um sistema que só libera acesso se **duas flags** estiverem true.
7. Converta um `if-else` de 6 níveis em `switch`.
8. Use operador ternário com interpolação de string.
9. Use flag para determinar quando encerrar um jogo de dados.
10. Refaça `if` gigante com dicionário (abstração).

🔷 Últimos desafios avançados

1. Crie condicional que verifica se três retas podem formar um triângulo.
2. Verifique se duas palavras são anagramas.
3. Controle de semáforo: verde = siga, amarelo = atenção, vermelho = pare.
4. Sistema de notas: calcule conceito final usando switch com faixas.
5. Verifique se um ponto `(x,y)` está dentro de um círculo.
6. Determine se horário está no expediente (8h–18h, exceto finais de semana).
7. Faça condicional para validar número de cartão de crédito (Luhn).
8. Valide se dois intervalos de datas colidem.
9. Cheque se senha não contém parte do email.
10. Refatore condicional gigante em Strategy Pattern (para aplicar abstração).