```
// ... usando um "modelo" e uma ou mais expressões de interpolação;
// Expressão de interpolação: variável cercada por um símbolo de chave de abertura e fechamento { };
// Cadeia de caracteres: se torna um modelo quando ele é prefixado pelo caractere $;

// string message = greeting + " " + firstName + "!";

// string message = $"{greeting} {firstName}!";
//-------------------------------------------------------------------------
// Etapa 1: usar a interpolação de cadeia de caracteres para combinar uma cadeia de caracteres literal e um valor variável

// string firstName = "Bob";
// string message = $"Hello {firstName}!";
// Console.WriteLine(message);
/*
    Hello Bob!
*/
//-------------------------------------------------------------------------
// Etapa 2: usar a interpolação de cadeia de caracteres com diversas cadeias de caracteres literais e variáveis

// string firstName = "Bob";
// string greeting = "Hello";
// string message = $"{greeting} {firstName}!";
// Console.WriteLine(message);
/*
    Hello Bob!
*/
//-------------------------------------------------------------------------
// Etapa 3: evitar variáveis intermediárias

// string firstName = "Bob";
// string greeting = "Hello";
// Console.WriteLine($"{greeting} {firstName}!");
/*
    Hello Bob!
*/
//-------------------------------------------------------------------------
// Etapa 4: combinar literais textuais e interpolação de cadeia de caracteres

// Usar um literal textual em seu modelo: usar o símbolo @ de prefixo literal textual e o símbolo $ de interpolação...
// ... de cadeia de caracteres juntos;

// string projectName = "First-Project";
// Console.WriteLine($@"C:\Output\{projectName}\Data");
/*
    C:\Output\First-Project\Data
*/
// A interpolação de cadeia de caracteres fornece uma melhoria na concatenação de cadeia de caracteres...
// ... reduzindo o número de caracteres necessários em algumas situações;

```