

Escrever seu primeiro código C#

Comece escrevendo pequenos exemplos de código para aprender os conceitos básicos da sintaxe C#.

Objetivos de aprendizagem

Após concluir este módulo, você poderá:

- Escrever suas primeiras linhas de código C#
- Usar duas técnicas diferentes para imprimir uma mensagem em um console de texto
- Diagnosticar erros ao digitar o código incorretamente
- Identificar diferentes elementos da sintaxe C#, como operadores, classes e métodos

Introdução

A linguagem de programação C# permite que você crie muitos tipos de aplicativos, como:

- Aplicativos de negócios para capturar, analisar e processar dados
- Aplicativos Web dinâmicos que podem ser acessados em um navegador da Web
- Jogos 2D e 3D
- Aplicativos financeiros e científicos
- Aplicativos seguros baseados em nuvem
- Aplicativos móveis

Mas como você começa a escrever um aplicativo?

Todos os aplicativos são compostos por muitas linhas de código que funcionam em conjunto para alcançar uma tarefa. De longe, a melhor maneira de aprender a codificar é escrever o máximo possível de códigos. Portanto, incentivamos que você escreva códigos em todos os exercícios deste e dos demais módulos deste roteiro de aprendizagem. Escrever os códigos sozinho em cada exercício e solucionar os pequenos desafios de codificação vai ajudar a acelerar o seu aprendizado. Você também precisa começar a aprender pequenos conceitos básicos e desenvolvê-los com a prática e a exploração contínuas.

Neste módulo, você vai:

- Gravar suas primeiras linhas de código C#.
- Use duas técnicas diferentes para imprimir uma mensagem na saída.
- Faça o diagnóstico dos erros quando o código estiver incorreto.
- Identificar diferentes elementos da sintaxe C#, como operadores, classes e métodos.

Ao final deste módulo, você conseguirá escrever código em C# para imprimir uma mensagem na saída padrão de um console, como o Terminal do Windows. Essas linhas de código oferecerão a você uma introdução à sintaxe C# e fornecerão informações úteis.

Exercício – Escrever seu primeiro código

Neste primeiro exercício prático, você usará o C# para imprimir uma frase consagrada do programador na saída padrão de um console.

Escrever sua primeira linha de código

Há uma tradição antiga entre os desenvolvedores de software de imprimir a frase "Olá, Mundo!" na janela de saída do console. Como você verá, você pode aprender muito sobre programação e a linguagem de programação C# com este simples exercício.

Digitar códigos no Editor do .NET

O Editor do .NET e o console de saída fornecem uma excelente experiência no navegador que é perfeita para essa abordagem do tutorial. O Editor do .NET está localizado no lado direito desta página da Web. O console de saída fica abaixo dele.

1. Insira este código exatamente como aparece no Editor do .NET, à direita:

```
C# Copiar

Console.WriteLine("Hello World!");
```

Você verá em breve uma explicação de como e por que ele funciona. Mas primeiro, você deve vê-lo em execução e conferir se inseriu tudo corretamente. Para fazer isso, você executará o código.

❗ Observação

Você pode querer selecionar **Copy** ou **Run** e ignorar toda a digitação. No entanto, há benefícios de digitar o código por conta própria. Inserir o código por conta própria reforça sua memória e sua compreensão e ajuda a obter insights que você não obterá de outra forma.

Executar seu primeiro código

1. Pressionar o botão Executar verde

O botão Executar verde executa duas tarefas:

- Ele compila o código em um formato executável que um computador possa entender.
- Ele executa o aplicativo compilado e, quando escrito corretamente, gera como saída "Hello World!".

Observar seus resultados

1. No console de saída, observe os resultados do seu código. Você verá a seguinte saída:

```
Output Copiar

Hello World!
```

O que fazer se uma mensagem de erro for exibida

Escrever o código C# é um exercício de precisão. Se você digitar um só caractere incorretamente, verá uma mensagem de erro na área de saída ao executar o código.

Por exemplo, se você inserir incorretamente um c em letra minúscula na palavra console da seguinte forma:

```
C# Copiar

console.WriteLine("Hello World!");
```

Você recebe a seguinte mensagem de erro:

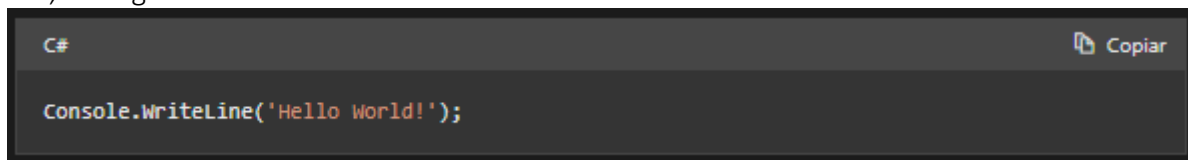
```
Output Copiar

(1,1): error CS0103: The name 'console' does not exist in the current context
```

A primeira parte (1,1) indica a linha e a coluna em que o erro ocorreu. Mas o que essa mensagem de erro significa?

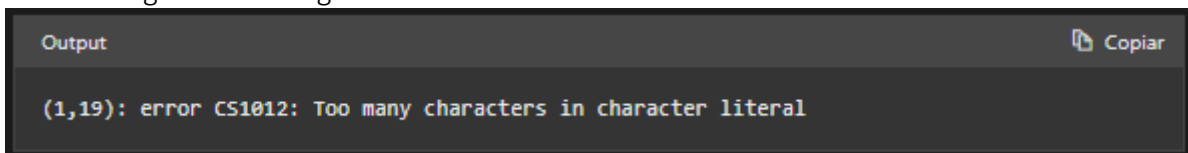
O C# é uma linguagem que diferencia maiúsculas de minúsculas, o que significa que o compilador C# considera as palavras `console` e `Console` tão diferentes quanto as palavras `cat` e `dog`. Às vezes, a mensagem de erro pode ser um pouco enganosa. Você precisará entender o verdadeiro motivo pelo qual o erro existe. Isso é possível com o aprendizado detalhado da sintaxe do C#.

Da mesma forma, se você usou aspas simples (') ao redor da cadeia de caracteres literal `Hello World!`, da seguinte maneira:



```
C#  
Console.WriteLine('Hello World!');
```

Você verá a seguinte mensagem de erro:



```
Output  
(1,19): error CS1012: Too many characters in character literal
```

Novamente, na linha 1, no caractere 19, encontramos o culpado. Você pode usar a mensagem como uma pista à medida que investiga o problema. Mas o que a mensagem de erro significa? O que é exatamente um "literal de caractere"? Posteriormente, você aprenderá mais sobre literais de vários tipos de dados (incluindo literais de caracteres). Por enquanto, apenas tenha cuidado ao inserir seu código.

Felizmente, os erros nunca são permanentes. Basta identificar o erro, corrigi-lo e executar o código novamente.

Se você receber um erro ao executar seu código, dedique um momento para analisá-lo com atenção. Examine cada caractere e verifique se inseriu essa linha de código com exatidão.

📌 Observação

O editor de códigos está monitorando constantemente o código que você escreve executando a pré-compilação para encontrar possíveis erros. Ele tentará ajudar você adicionando linhas onduladas vermelhas abaixo do código que vai gerar erro.

Erros comuns cometidos por programadores inexperientes:

- Inserir letras minúsculas em vez de colocar C em maiúsculas em `Console` ou as letras W ou L em `WriteLine`.
- Inserir uma vírgula em vez de um ponto entre `Console` e `WriteLine`.
- Esquecer de usar aspas duplas ou colocar a frase `Hello World!` entre aspas simples.
- Esquecer de usar o ponto e vírgula no final do comando.

Cada um desses erros impede que o código seja compilado com êxito.

O editor de código realça erros de pré-compilação para ajudar você a identificar e corrigir erros mais facilmente ao desenvolver seu código. Você pode encarar isso como um verificador ortográfico que ajuda a corrigir erros gramaticais ou ortográficos em um documento de texto.

Supondo que você teve êxito nas etapas anteriores, vamos prosseguir.

Exibir uma nova mensagem

Nesta tarefa, você comentará a linha de código anterior e, em seguida, adicionará novas linhas de código no Editor do .NET para imprimir uma nova mensagem

1. Modifique o código que você escreveu inserindo como prefixo a chave de comentário, que são duas barras consecutivas `//`:

```
C# Copiar

// Console.WriteLine("Hello World!");
```

1. Você pode criar um comentário de código adicionando duas barras consecutivas `//` como prefixo da linha de código. Esse prefixo instrui o compilador a ignorar todas as instruções daquela linha específica.

Os comentários de código são úteis quando você ainda não está pronto para excluir um trecho de código, mas deseja ignorá-lo por enquanto. Você também pode usar comentários de código para adicionar mensagens a si mesmo ou a outras pessoas que possam ler seu código posteriormente, lembrando a função daquele trecho de código, por exemplo.

2. Adicione novas linhas de código que correspondam ao seguinte snippet de código:

```
C# Copiar

Console.Write("Congratulations!");
Console.Write(" ");
Console.Write("You wrote your first lines of code.");
```

Pressione o botão Executar verde novamente. Desta vez, você deverá ver a seguinte saída.

```
Output Copiar

Congratulations! You wrote your first lines of code.
```

A diferença entre `Console.Write` e `Console.WriteLine`

As três novas linhas de código que você adicionou demonstraram a diferença entre os métodos `Console.WriteLine()` e `Console.Write()`.

Para imprimir uma mensagem inteira no console de saída, você usou a primeira técnica, `Console.WriteLine()`. Ao final da linha, ela adicionou uma alimentação de modo semelhante à criação de uma linha de texto pressionando Enter ou Return.

Para imprimir no console de saída, mas sem adicionar um caractere de nova linha ao final da frase, você usou a segunda técnica, `Console.Write()`. Assim, a próxima chamada a `Console.Write()` imprime outra mensagem na mesma linha.

Parabéns por escrever suas primeiras linhas de código!

Saber como isso funciona

Para entender como seu código funciona, você precisa relembrar o que é uma linguagem de programação. Pense sobre como seu código comunica comandos ao computador.

O que é uma linguagem de programação?

Linguagens de programação como o C# permitem que você escreva instruções a serem executadas pelo computador. Cada linguagem de programação tem uma sintaxe própria, mas após aprender a primeira, ao tentar aprender outra você perceberá rapidamente que todas elas têm muitos conceitos semelhantes. O trabalho de uma linguagem de programação é permitir que uma pessoa expresse sua intenção para o computador, de maneira compreensível e legível por humanos. As

instruções que você escreve em uma linguagem de programação são chamadas de "código-fonte" ou apenas de "código". Os desenvolvedores de software escrevem código. Neste ponto, um desenvolvedor pode atualizar e alterar códigos, mas o computador não pode entendê-los diretamente. Primeiro, o código precisa ser compilado em um formato que o computador possa entender.

O que é compilação?

Um programa especial chamado **compilador** converte o código-fonte em um formato diferente que a CPU (unidade de processamento central) do computador pode executar. Quando você usou o botão **Executar** verde na unidade anterior, o código escrito foi compilado pela primeira vez e executado.

Por que o código precisa ser compilado? Embora a maioria das linguagens de programação pareça enigmática no início, elas podem ser mais facilmente compreendidas por humanos do que a linguagem preferencial do computador. A CPU entende as instruções expressas ativando ou desativando milhares ou milhões de pequenos comutadores. Os compiladores conectam esses dois mundos convertendo as instruções legíveis por humanos em um conjunto de instruções compreensível para o computador.

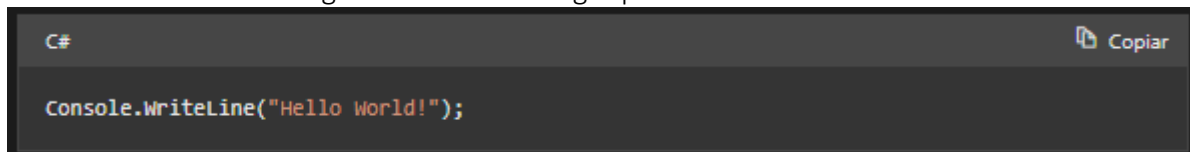
O que é sintaxe?

As regras para escrever código em C# são chamadas de sintaxe. Assim como as linguagens humanas têm regras sobre pontuação e estrutura de frases, as linguagens de programação de computador também têm regras. Essas regras definem as palavras-chave e os operadores de C# e como eles são combinados para formar programas.

Ao escrever um código no Editor do .NET, você pode ter notado alterações sutis nas cores de diferentes palavras e símbolos. O realce de sintaxe é um recurso útil que você começará a usar para identificar facilmente erros em seu código que não estão em conformidade com as regras de sintaxe do C#.

Como o código funcionou?

Vamos nos concentrar na seguinte linha de código que você escreveu:

A screenshot of a code editor window with a dark background. The title bar at the top shows 'C#' on the left and a 'Copiar' button on the right. The main area contains a single line of C# code: `Console.WriteLine("Hello World!");`. The text is color-coded: 'Console' is blue, 'WriteLine' is orange, and the string 'Hello World!' is enclosed in red double quotes.

Ao executar o seu código, você notou que a mensagem Hello World! foi impressa no console de saída. Quando a frase é colocada entre aspas duplas no código C#, ela é chamada de **cadeia de caracteres literal**. Em outras palavras, literalmente, você queria enviar os caracteres H, e, l, l, o e assim por diante à saída.

A parte Console é chamada de **classe**. As classes "contêm" métodos; ou também podemos dizer que os métodos residem nas classes. Para visitar o método, é necessário saber em qual classe ele está. Por enquanto, pense em uma classe como uma forma de representar um objeto. Nesse caso, todos os métodos que operam no console de saída são definidos dentro da classe Console.

Também há um ponto que separa o nome da classe Console e o nome do método WriteLine(). O ponto é o operador de acesso a membro. Em outras palavras, o ponto é a forma como você "navega" da classe para um dos métodos dela.

A parte `WriteLine()` é chamada de **método**. Você sempre pode identificar um método porque, após ele, há um conjunto de parênteses. Cada método tem um trabalho. O trabalho do método `WriteLine()` é gravar uma linha de dados no console de saída. Os dados impressos são enviados entre os parênteses de abertura e de fechamento como um parâmetro de entrada. Alguns métodos precisam de parâmetros de entrada, enquanto outros não. Mas se você quiser invocar um método, sempre precisará usar os parênteses após o nome dele. Os parênteses são conhecidos como o operador de invocação de método.

Por fim, o ponto e vírgula é o final do operador de instrução. Uma **instrução** é uma instrução completa em C#. O ponto e vírgula indica ao compilador que você terminou de inserir o comando. Não se preocupe se todas essas ideias e esses termos não fizerem sentido. Por enquanto, para imprimir uma mensagem no console de saída, você só precisa se lembrar de:

- Use `Console.WriteLine("Your message here");`.
- Coloque `Console`, `Write` e `Line` em maiúsculas
- Usar a pontuação correta, pois ela tem uma função especial em C#
- Se você cometer um erro, identifique-o, corrija-o e execute novamente

Dica

Crie uma folha de referências para você mesmo até ter memorizado determinados comandos chave.

Entender o fluxo de execução

Além disso, é importante entender o fluxo de execução. Em outras palavras, as instruções de código foram executadas em ordem, uma linha por vez, até não haver mais instruções a serem executadas. Algumas instruções exigirão que a CPU aguarde antes de continuar. Outras instruções podem ser usadas para alterar o fluxo de execução.

Agora, vamos testar o que você aprendeu. Cada módulo apresenta um desafio simples e, se você tiver dificuldades, forneceremos uma solução. Na próxima unidade, você terá a chance de escrever um pouco de código C# por conta própria.

Verificar seu conhecimento

1. Qual é a diferença entre `Console.Write` e `Console.WriteLine`?

- ☐ `Console.Write` imprime a saída em uma nova linha.
- ☐ `Console.WriteLine` imprime a saída em uma nova linha.
- ☐ `Console.WriteLine` acrescenta uma nova linha após a saída.

Concluir o desafio

Os desafios de código desses módulos reforçarão o que você aprendeu e ajudarão você a ganhar confiança antes de continuar.

Desafio: escrever o código no Editor do .NET para exibir duas mensagens

1. Selecione todo o código no Editor do .NET e pressione a tecla `Delete` ou `Backspace` para excluir.
2. Escreva um código que produz a seguinte saída:

```
Output Copiar  
  
This is the first line.  
This is the second line.
```

1. Na unidade anterior, você aprendeu a exibir uma mensagem em apenas uma linha de código e exibir uma mensagem usando várias linhas de código. Use as duas técnicas para esse desafio. Não importa qual técnica você aplicará a qual linha e não importa por quantas maneiras você dividirá uma das mensagens em várias linhas de código. Isso fica a seu critério. Independentemente de como você fizer isso, o código deverá produzir a saída especificada.

Se você tiver dificuldades e precisar espiar a solução ou mesmo se terminar com sucesso, prossiga até a próxima unidade para ver uma solução para esse desafio.

Revisar a solução

```
C# Copiar  
  
Console.WriteLine("This is the first line.");  
  
Console.Write("This is ");  
Console.Write("the second ");  
Console.Write("line.");
```

O código a seguir é uma possível solução para o desafio da unidade anterior.

Esse código é apenas *uma solução possível*, entre muitas maneiras possíveis de obter o mesmo resultado. No entanto, você deve ter usado os métodos `Console.WriteLine()` e `Console.Write(String)` para produzir a saída desejada.

```
Output Copiar  
  
This is the first line.  
This is the second line.
```

Se você conseguiu, parabéns! Prossiga para a próxima unidade para uma verificação de conhecimentos.

📌 Importante

Se você teve dificuldades para conduzir o desafio, reveja as unidades anteriores antes de continuar.

Verificar seus conhecimentos

1. Qual é o trabalho principal do compilador?

- ☐ O compilador localiza principalmente erros de ortografia no código.
- ☐ O compilador executa principalmente o código.
- ☐ O compilador converte principalmente o código em um formato executável que o computador possa entender.

2. Qual das instruções a seguir é verdadeira sobre o C#?

- ☐ O C# não diferencia maiúsculas de minúsculas.
- ☐ `Console` é um método e `WriteLine()` é uma classe.
- ☐ Aspas duplas são usadas para criar uma cadeia de caracteres literal.

3.O que há de errado com esta linha de código? `Console.WriteLine("What is wrong with me?")`

- ☐ O L em WriteLine deve estar em letra minúscula.
- ☐ Falta um ponto e vírgula no final
- ☐ A cadeia de caracteres deve usar aspas simples.

Resumo

Seu objetivo era escrever códigos que exibissem mensagens simples no console de saída para se familiarizar com a sintaxe. Você escreveu suas primeiras linhas de código usando a sintaxe básica do C#. Você aprendeu duas técnicas para exibir dados de cadeias de caracteres literais no console. Você também aprendeu o que procurar ao se deparar com um erro no código. Por fim, você identificou elementos da sintaxe do C# como classes e métodos e a finalidade de vários símbolos especiais conhecidos como operadores. Você executou as primeiras etapas para criar aplicativos mais sofisticados.