

Olá, estudante! Seja muito bem-vindo ao curso de programação em C#, módulo **“Escrever o seu primeiro Código C#”.**

Antes de começar os seus estudos é importante saber que em cada módulo do curso você terá acesso a dois materiais de estudos. O primeiro sempre será a aula em vídeo, onde você aprenderá a parte prática do conteúdo abordado. O segundo é uma pequena apostila com as mesmas informações, porém nela você poderá ter acesso aos códigos para testar em seu próprio computador sem precisar pausar e copiar os códigos apresentados em vídeo e também aos links de acesso, programas e sites.

Para otimizar sua prática, utilize o editor .NET que está disponível em: <https://dotnet.microsoft.com/en-us/platform/try-dotnet>.

**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 começar?

De longe, a melhor maneira de aprender a codificar é *escrever* o máximo possível de códigos. Portanto, incentivamos você a inserir código com os exercícios deste módulo e os outros deste roteiro de aprendizagem. Inserir o código você mesmo em cada exercício e solucionar pequenos desafios de codificação acelerará seu aprendizado.

Além disso, você precisa começar a aprender pequenos conceitos básicos e desenvolvê-los com a prática e a exploração contínuas.

Neste módulo, você vai:

● 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

Ao final deste módulo, você conseguirá escrever código C# para exibir uma mensagem em um console de texto, como o prompt de comando do Windows. Essas linhas de código oferecerão a você uma introdução à sintaxe C# e fornecerão insights úteis imediatamente. Na verdade, este módulo apresentará muitas novas ideias, que serão expandidas em outros módulos.

**Exercício - “Olá, Mundo!”**

Neste primeiro exercício prático, você usará o C# para imprimir uma frase consagrada do programador na Janela de Saída.

**Escrever sua primeira linha de código**

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

**Etapa 1: inserir o código no Editor do .NET**

O Editor do .NET e o painel de saída fornecem uma ótima 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 painel de saída está abaixo dele.

Mais adiante, você aprenderá a usar o Visual Studio Code ou o IDE completo do Visual Studio para criar exemplos de código maiores e mais interessantes – até mesmo aplicativos inteiros.

Insira este código no Editor do .NET à direita:



Explicaremos como e por que isso funciona em breve. Mas primeiro, você deve vê-lo em execução e conferir se não o inseriu incorretamente. Para fazer isso, você executará o código.

**Observação**

Você pode querer selecionar Copy ou Run e pular toda a digitação. No entanto, incentivamos você a inserir essa linha de código por conta própria. Inserir o código por conta própria reforça sua capacidade de memória e ajuda a obter insights que você não obteria de outra forma.

**Etapa 2: 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 gera o comando desejado.

**Etapa 3: Na Janela de Saída, observe o resultado**

A saída a seguir será exibida.



**O que fazer se uma mensagem de erro for exibida**

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

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

A seguinte mensagem de erro será exibida:



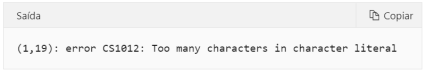
A primeira parte (1,1) indica a linha e a coluna em que o erro ocorreu. Mas o que essa mensagem de erro significa?

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 entre a cadeia de caracteres literal Hello World! desta maneira:



A seguinte mensagem de erro será exibida:



Novamente, a linha 1, o caractere 19 nos indica 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"? Embora vamos descrever literais de vários tipos de dados (incluindo literais de caracteres) em outro módulo, nosso conselho por enquanto é ter cuidado ao inserir o código.

Felizmente, os erros nunca são permanentes. Você simplesmente identifica o erro, corrige-o e executa novamente o aplicativo.

Se você receber um erro ao executar o código, reserve um momento, examine cada caractere e confira se você inseriu essa linha de código de maneira exata.

**Observação**

O editor de códigos está monitorando constantemente o código que você escreve executando a pré-compilação para encontrar possíveis erros. Ele tentará ajudá-lo a adicionar linhas onduladas vermelhas abaixo do código que produzirá um 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.

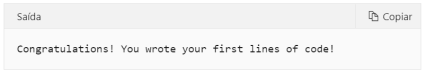
Cada um desses erros impede que o código seja compilado com êxito. Supondo que você teve êxito nas etapas anteriores, vamos prosseguir.

**Etapa 4: comentar a linha de código anterior e, em seguida, adicionar novas linhas de código no Editor do .NET para imprimir uma nova mensagem**

Modifique o código que você escreveu adicionando como prefixo o símbolo de comentário de código (ou seja, duas barras // ) e adicione novas linhas de código para corresponder ao seguinte snippet:



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



Você cria um comentário de código adicionando duas barras // como prefixo a uma linha de código. Isso 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 os comentários de código a fim de adicionar mensagens para si mesmo lembrando o que faz aquele trecho de código específico. Falaremos sobre comentários de código em um módulo futuro.

As três novas linhas de código que você adicionou demonstraram a diferença entre os métodos Console.WriteLine() e Console.Write(). Naturalmente, as novas linhas de código que você adicionou poderiam ser reduzidas com facilidade a uma linha de código, como feito na etapa 1. No entanto, você queria aprender uma segunda técnica para imprimir uma mensagem no console de saída.

Para imprimir uma mensagem inteira no console de saída, a primeira técnica usou 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 a impressão no console de saída, mas no final, sem adicionar uma alimentação de linha, a segunda técnica usou Console.Write(). Assim, a próxima chamada a Console.Write() imprime uma mensagem adicional na mesma linha.

Além das duas técnicas para a impressão de mensagens no console de saída, há mais a aprender com este exercício. O que todas essas palavras e esses símbolos significam?

**Como ele funciona**

Para entender como o código funciona, você parar um pouco e pensar no que é uma linguagem de programação e como ela comunica os comandos ao computador.

**O que é uma linguagem de programação?**

Linguagens de programação como o C# permitem que você escreva instruções que você deseja que sejam executadas pelo computador. Cada linguagem de programação tem uma sintaxe diferente, mas depois de aprender sua primeira linguagem de programação e, em seguida, tentar aprender uma segunda, você perceberá rapidamente que todas elas têm muitas ideias semelhantes. O trabalho de uma linguagem de programação é permitir que um humano escreva instruções de maneira compreensível e legível por humanos. As instruções que você escreve em uma linguagem de programação são chamadas de "código-fonte" ou apenas "código".

Neste ponto, o código pode ser atualizado e alterado por um desenvolvedor de software, mas o computador não pode entender o código. Primeiro, ele 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 é executável pela CPU do computador. Quando você usou o botão Executar verde na unidade anterior, o código escrito foi compilado pela primeira vez e executado.

Por que precisamos compilar nosso código? Embora a maioria das linguagens de programação pareça criptografada a princípio, elas podem ser compreendidas com mais facilidade pelos humanos do que pela linguagem *preferencial* do computador, que é expressa por meio da ativação ou da desativação de 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?**

A sintaxe de uma linguagem de programação inclui as palavras-chave, os operadores (caracteres especiais do teclado, como ponto e vírgula ou parênteses) e outras regras gramaticais impostas pelo compilador. As linhas de código digitadas seguiram dezenas de regras de sintaxe diferentes e usaram, pelo menos, quatro operadores diferentes. Há muito a aprender, mas felizmente cada conceito é simples. Não desista! Você aprenderá tudo isso.

Ao inserir o código no Editor do .NET, talvez você tenha observado alterações sutis na cor de diferentes palavras e símbolos. O realce de sintaxe é um recurso útil que você começará a usar para identificar com facilidade os erros no código que não estão em conformidade com as regras da sintaxe do C#. Na verdade, uma versão semelhante (e ainda mais robusta) desse recurso está disponível no Visual Studio Code e no IDE completo do Visual Studio.

**Como o código funcionou?**

Vamos nos concentrar na linha de código a seguir que você escreveu. 

Quando você executou o código, viu que a mensagem Hello World! foi impressa no painel 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, queríamos que os caracteres H, e, l, l, o e assim por diante fossem enviados para a saída. Você aprenderá mais sobre cadeias de caracteres literais no módulo intitulado "Armazenar e recuperar dados usando valores de literais e variáveis no C#".

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 na Janela 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, outros não. Mas se você quiser invocar um método, sempre precisará usar os parênteses após o nome do método. Os parênteses são conhecidos como o *operador de invocação de método*. Você aprenderá mais sobre como chamar métodos no módulo intitulado "Aproveitar bibliotecas de funcionalidade chamando métodos na biblioteca de classes .NET no C#".

A parte Console é chamada de classe. As classes são "proprietárias" de métodos ou talvez uma maneira melhor de dizer isso é que os métodos residem em uma classe. Para visitar o método, é necessário saber em qual classe ele está. Por enquanto, considere uma classe como uma maneira de armazenar e organizar todos os métodos que fazem coisas semelhantes. Nesse caso, todos os métodos que operam no painel de Saída são definidos na 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.

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 informa ao compilador que terminamos de inserir o comando.

Não se preocupe se todas essas ideias e esses termos não fizerem sentido. Por enquanto, apenas lembre-se de que se desejar imprimir uma mensagem em uma Janela de Saída como um console:

● Use Console.WriteLine("Your message here");.

● Coloque Console, Write e Line em maiúsculas

● Use a *pontuação* correta, pois ela tem uma função especial no C#

● Se você cometer um erro, apenas identifique-o, corrija-o e execute novamente o aplicativo... não tem como dar errado

**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. Você aprenderá mais sobre essas situações especiais enquanto aprende mais sobre a sintaxe C# e os métodos na biblioteca de classes .NET.

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.

**Desafio**

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

**Etapa 1: Excluir todo o código no Editor do .NET do exercício anterior**

Selecione todo o código no Editor do .NET e pressione a tecla Delete ou Backspace para excluir.

**Etapa 2: Escrever o código no Editor do .NET para exibir duas mensagens**

Use as duas técnicas diferentes para imprimir as duas linhas da saída. Em outras palavras, 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.

Não importa como você fará isso; o código deverá produzir a saída a seguir. 

Se você não conseguir avançar e precisar ver a solução ou terminar com êxito, prossiga para ver uma solução para este desafio.

**Solução**

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



Esse código é meramente "*uma possível solução*", porque não especificamos qual técnica deve ser aplicada a qual linha de saída. No entanto, você deve ter usado os métodos Console.WriteLine() e Console.Write() para produzir a saída desejada.



**Importante**

Se você teve dificuldades para concluir o desafio, talvez seja melhor rever as unidades anteriores antes de continuar. Todas as novas ideias que descrevemos em outros módulos dependem de sua compreensão das ideias apresentadas neste módulo.   
  
  
**Resumo**

Nossa meta é escrever um código que exiba mensagens simples em um console de saída e, ao fazer isso, criar familiaridade com a sintaxe. Escrevemos nossas primeiras

linhas de código usando a sintaxe C# básica. Aprendemos duas técnicas para exibir dados de cadeia de caracteres literal no console. Aprendemos o que procurar quando vemos um erro no código. Por fim, identificamos os elementos da sintaxe C# como classes e métodos e a finalidade de vários símbolos especiais conhecidos como operadores.

Parabéns por realizar suas primeiras etapas para criar aplicativos mais sofisticados. **Saiba mais com uma série de vídeos do Canal 9**

● C# 101

● C# Avançado