**Main() 和命令行参数（C# 编程指南）**

**Main**方法是 C# 控制台应用程序或窗口应用程序的入口点。（库和服务不要求将 **Main** 方法作为入口点。）应用程序启动时，**Main** 方法是第一个调用的方法。

C# 程序中只能有一个入口点。如果您有多个类都包含 **Main** 方法，则必须使用 **/main** 编译器选项编译您的程序，以指定用作入口点的 **Main** 方法。有关更多信息，请参见[/main (C# Compiler Options)](https://msdn.microsoft.com/zh-cn/library/x3eht538.aspx)。

class TestClass

{

static void Main(string[] args)

{

// Display the number of command line arguments:

System.Console.WriteLine(args.Length);

}

}

**概述**

* **Main** 方法是 .exe 程序的入口点，程序控制流在该处开始和结束。
* **Main** 在类或结构内声明。 **Main** 必须是 [静态](https://msdn.microsoft.com/zh-cn/library/98f28cdx.aspx)，且不应该是 [公开](https://msdn.microsoft.com/zh-cn/library/yzh058ae.aspx)。（在前面的示例中，它接受默认访问级别 [private](https://msdn.microsoft.com/zh-cn/library/st6sy9xe.aspx)。）但不要求封闭类或结构是静态的。
* **Main** 的返回类型有两种：**void** 或 **int**。
* 所声明的 **Main** 方法可以具有包含命令行实参的 string[] 形参，也可以不具有这样的形参。使用 Visual Studio 创建 Windows 窗体应用程序时，可以手动添加形参，也可以使用 [Environment](https://msdn.microsoft.com/zh-cn/library/system.environment.aspx) 类获取命令行实参。形参读取为零索引的命令行参数。与 C 和 C++ 不同，程序的名称视为第一个命令行参数。

**命令行参数（C# 编程指南）**

通过以下方式之一定义方法，可以将参数发送至 **Main** 方法。

static int Main(string[] args)

static void Main(string[] args)

| **https://wizardforcel.gitbooks.io/msdn-csharp/content/img/note.jpg 注意** |
| --- |
| 若要在 Windows 窗体应用程序中的 **Main** 方法中启用命令行参数，必须手动修改 program.cs 中 **Main** 的签名。Windows 窗体设计器生成的代码创建没有输入参数的 **Main**。也可以使用 [Environment.CommandLine](https://msdn.microsoft.com/zh-cn/library/system.environment.commandline.aspx) 或 [Environment.GetCommandLineArgs](https://msdn.microsoft.com/zh-cn/library/system.environment.getcommandlineargs.aspx) 从控制台或 Windows 应用程序中的任何位置访问命令行参数。 |

**Main** 方法的参数是表示命令行参数的 [String](https://msdn.microsoft.com/zh-cn/library/system.string.aspx) 数组。一般是通过测试 Length 属性来确定参数是否存在，例如：

if (args.Length == 0)

{

System.Console.WriteLine("Please enter a numeric argument.");

return 1;

}

还可以使用 [Convert](https://msdn.microsoft.com/zh-cn/library/system.convert.aspx) 类或 **Parse** 方法将字符串参数转换为数值类型。例如，下面的语句使用 [Parse](https://msdn.microsoft.com/zh-cn/library/3b6b4bx3.aspx) 方法将 string 转换为 long 数字：

long num = Int64.Parse(args[0]);

也可以使用别名为 Int64 的 C# 类型 long：

long num = long.Parse(args[0]);

还可以使用 Convert 类的方法 ToInt64 完成同样的工作：

long num = Convert.ToInt64(s);

有关更多信息，请参见[Parse](https://msdn.microsoft.com/zh-cn/library/3b6b4bx3.aspx)和[Convert](https://msdn.microsoft.com/zh-cn/library/system.convert.aspx)。

下面的示例演示如何在控制台应用程序中使用命令行参数。应用程序在运行时采用一个参数，将该参数转换为整数，并计算该数的阶乘。如果没有提供参数，则应用程序发出一条消息来解释程序的正确用法。

若要根据命令提示编译并运行应用程序，请执行以下步骤：

1. 将以下代码粘贴到任何文本编辑器中，并将文件保存为名为 Factorial.cs 的文本文件。
2. //Add a using directive for System if the directive isn't already present.
3. public class Functions
4. {
5. public static long Factorial(int n)
6. {
7. // Test for invalid input
8. if ((n &lt; 0) || (n &gt; 20))
9. {
10. return -1;
11. }
12. // Calculate the factorial iteratively rather than recursively:
13. long tempResult = 1;
14. for (int i = 1; i &lt;= n; i++)
15. {
16. tempResult \*= i;
17. }
18. return tempResult;
19. }
20. }
21. class MainClass
22. {
23. static int Main(string[] args)
24. {
25. // Test if input arguments were supplied:
26. if (args.Length == 0)
27. {
28. System.Console.WriteLine("Please enter a numeric argument.");
29. System.Console.WriteLine("Usage: Factorial <num>");
30. return 1;
31. }
32. // Try to convert the input arguments to numbers. This will throw
33. // an exception if the argument is not a number.
34. // num = int.Parse(args[0]);
35. int num;
36. bool test = int.TryParse(args[0], out num);
37. if (test == false)
38. {
39. System.Console.WriteLine("Please enter a numeric argument.");
40. System.Console.WriteLine("Usage: Factorial <num>");
41. return 1;
42. }
43. // Calculate factorial.
44. long result = Functions.Factorial(num);
45. // Print result.
46. if (result == -1)
47. System.Console.WriteLine("Input must be >= 0 and <= 20.");
48. else
49. System.Console.WriteLine("The Factorial of {0} is {1}.", num, result);
50. return 0;
51. }
52. }
53. // If 3 is entered on command line, the
54. // output reads: The factorial of 3 is 6.
55. 从“开始”屏幕或“开始”菜单中，打开 Visual Studio“开发人员命令提示”窗口，然后导航到包含您刚创建的文件的文件夹。
56. 若要编译应用程序，请输入下面的命令。

csc Factorial.cs

如果您的应用程序中有没有编译错误，则将创建名为 Factorial.exe 的可执行文件。

1. 输入以下命令来计算 3 的阶乘：

Factorial 3

1. 此命令将生成以下输出：The factorial of 3 is 6.

| **https://wizardforcel.gitbooks.io/msdn-csharp/content/img/note.jpg 注意** |
| --- |
| 在 Visual Studio 中运行应用程序时，可以在[“项目设计器”->“调试”页](https://msdn.microsoft.com/zh-cn/library/2wcdezs5.aspx)中指定命令行参数。 |

有关如何使用命令行参数的更多示例，请参见[如何：使用命令行创建和使用程序集（C# 和 Visual Basic）](https://msdn.microsoft.com/zh-cn/library/3707x96z.aspx)。

**注意:在c和c++中命令行参数包括应用程序本身但是在c#中命令行参数不包括应用程序本身**

**实例1**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Csharpbasic\_cmd\_args  {  internal class Program  {  static void Main(string[] args)  {  foreach (var item in args)  {  Console.WriteLine(item);  }  }  }  } |

### 效果

|  |
| --- |
|  |

# 显示命令行参数（C# 编程指南）

可以通过 Main 的可选参数来访问通过命令行提供给可执行文件的参数。参数以字符串数组的形式提供。数组的每个元素都包含一个参数。参数之间的空白被移除。例如，下面是对一个假想的可执行文件的命令行调用：

| **命令行输入** | **传递给 Main 的字符串数组** |
| --- | --- |
| **executable.exe a b c** | "a""b"“c” |
| **executable.exe one two** | "one""two" |
| **executable.exe "one two" three** | "one two""three" |

| **https://wizardforcel.gitbooks.io/msdn-csharp/content/img/note.jpg 注意** |
| --- |
| 在 Visual Studio 中运行应用程序时，可以在[“项目设计器”->“调试”页](https://msdn.microsoft.com/zh-cn/library/2wcdezs5.aspx)中指定命令行参数。 |

本示例显示了传递给命令行应用程序的命令行参数。显示的输出对应于上表中的第一项。

class CommandLine

{

static void Main(string[] args)

{

// The Length property provides the number of array elements

System.Console.WriteLine("parameter count = {0}", args.Length);

for (int i = 0; i < args.Length; i++)

{

System.Console.WriteLine("Arg[{0}] = [{1}]", i, args[i]);

}

}

}

/\* Output (assumes 3 cmd line args):

parameter count = 3

Arg[0] = [a]

Arg[1] = [b]

Arg[2] = [c]

\*/

# 使用 foreach 访问命令行参数（C# 编程指南）

循环访问数组的另一种方法是使用 [foreach](https://msdn.microsoft.com/zh-cn/library/ttw7t8t6.aspx) 语句，如下面的示例所示。 **foreach** 语句可以用于循环访问数组、.NET Framework 集合类或任何实现 [IEnumerable](https://msdn.microsoft.com/zh-cn/library/system.collections.ienumerable.aspx) 接口的类或结构。

| **https://wizardforcel.gitbooks.io/msdn-csharp/content/img/note.jpg 注意** |
| --- |
| 在 Visual Studio 中运行应用程序时，可以在[“项目设计器”->“调试”页](https://msdn.microsoft.com/zh-cn/library/2wcdezs5.aspx)中指定命令行参数。 |

下面的示例演示如何使用 **foreach** 输出命令行参数。

// arguments: John Paul Mary

class CommandLine2

{

static void Main(string[] args)

{

System.Console.WriteLine("Number of command line parameters = {0}", args.Length);

foreach (string s in args)

{

System.Console.WriteLine(s);

}

}

}

/\* Output:

Number of command line parameters = 3

John

Paul

Mary

\*/

# Main() 返回值（C# 编程指南）

Main 方法可以返回 **void**：

static void Main()

{

//...

}

它还可以返回 **int**：

static int Main()

{

//...

return 0;

}

如果不使用 Main 的返回值，则返回 **void** 可以稍微简化代码。但是，如果返回整数，则程序可以与调用该可执行文件的其他程序或脚本交流状态信息。下面的示例演示如何访问 Main 的返回值。

此示例使用一个批处理文件来运行程序，并测试 Main 函数的返回值。在 Windows 中执行程序时，Main 函数返回的任何值都将存储在名为 ERRORLEVEL 的环境变量中。通过检查 ERRORLEVEL 变量，批处理文件可以确定执行结果。通常，返回值为零指示执行成功。下面是一个简单程序示例，从 Main 函数返回零。零指示程序成功运行。请将该程序保存为 MainReturnValTest.cs。

// Save this program as MainReturnValTest.cs.

class MainReturnValTest

{

static int Main()

{

//...

return 0;

}

}

此示例使用了批处理文件，因此最好在命令提示中编译这段代码。请按照[How to: Set Environment Variables for the Visual Studio Command Line](https://msdn.microsoft.com/zh-cn/library/1700bbwd.aspx)中的说明启用命令行生成，或者使用 Visual Studio 命令提示（可通过**“开始”**菜单中的**“Visual Studio Tools”**访问）。在命令提示符下，定位到保存程序的文件夹。下面的命令编译 MainReturnValTest.cs，生成可执行文件 MainReturnValTest.exe。

csc MainReturnValTest.cs

接下来，创建一个批处理文件，运行 MainReturnValTest.exe 并显示结果。将下面的代码粘贴到文本文件中，将该文件另存为 test.bat，保存到包含 MainReturnValTest.cs 和 MainReturnValTest.exe 的文件夹中。在命令提示符下，键入 test，运行该批处理文件。

因为代码返回零，所以该批处理文件会报告成功。但是，如果将 MainReturnValTest.cs 更改为返回非零值，然后重新编译程序，则批处理文件的后续执行将报告失败。

rem test.bat

@echo off

MainReturnValTest

@if "%ERRORLEVEL%" == "0" goto good

:fail

echo Execution Failed

echo return value = %ERRORLEVEL%

goto end

:good

echo Execution succeeded

echo Return value = %ERRORLEVEL%

goto end

:end

## 示例输出

Execution succeeded

Return value = 0