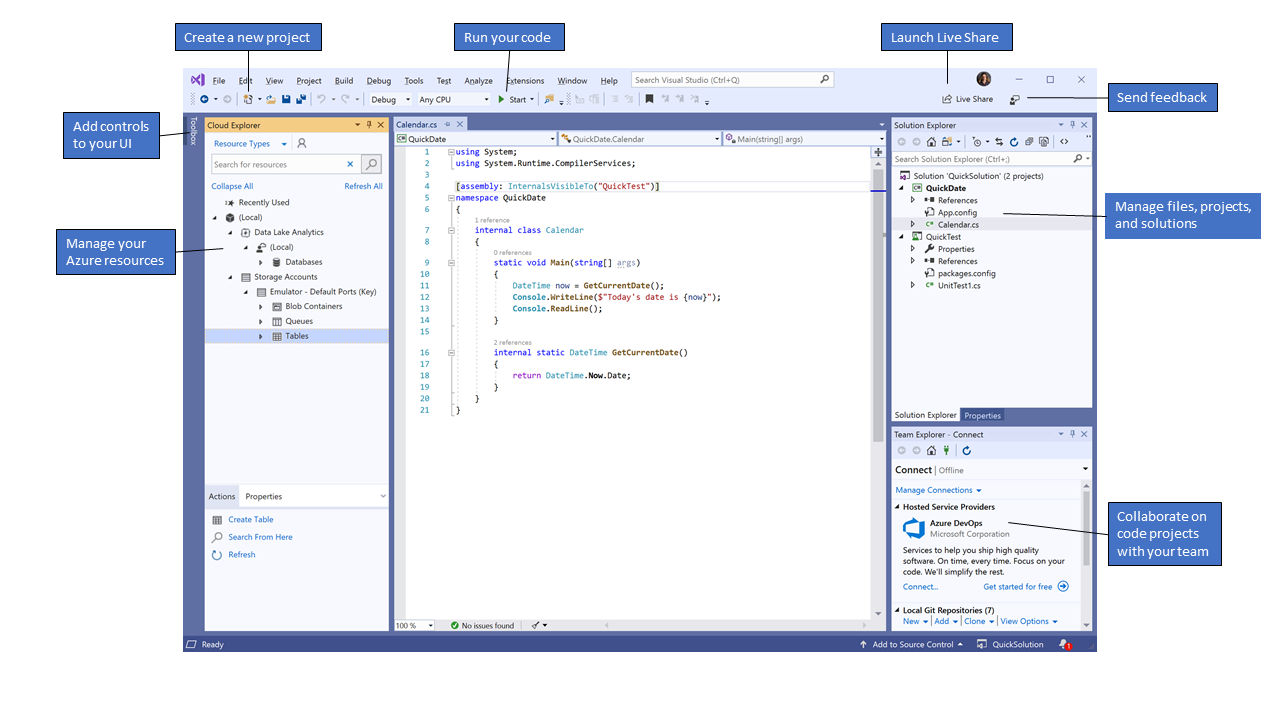
# 关于Visual Studio IDE[[1]](#footnote-1)

Visual Studio 集成开发环境是一种可用于编辑、调试并生成代码，然后发布应用 。 集成开发环境 (IDE) 是一个功能丰富的程序，可用于软件开发的许多方面。 除了大多数 IDE 提供的标准编辑器和调试器之外，Visual Studio 还包括编译器、代码完成工具、图形设计器和许多其他功能，以简化软件开发过程。



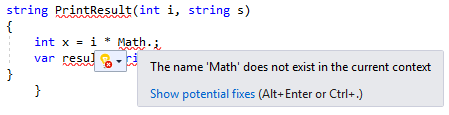
上图重要的有

1. **解决方案资源管理器（右上方）用于管理解决方案(Solution)和项目(projects)。**
2. **代码编辑器**（中间）**:** Visual Studio 编辑器提供了许多功能，可方便你更加轻松地编写和管理代码和文本。 可通过使用大纲显示来展开和折叠不同的代码块。 可通过使用 IntelliSense、“对象浏览器” 以及调用层次结构，了解有关代码的详细信息。 可使用“转到” 、“转到定义” 和“查找所有引用” 等功能查找代码。 可以将插入代码块和代码片段，并且可以通过“使用时生成” 等功能生成代码。
3. **团队资源管理器**（右下方）：利用版本控制技术(Git)让你跟踪团队开发的工作并且与他人分享代码。

## 提高编码效率的特色功能

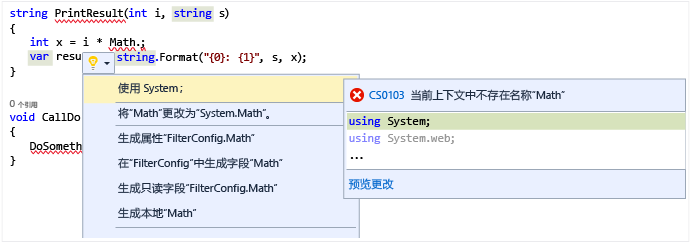
### 1.波形曲线和[快速操作](https://docs.microsoft.com/zh-cn/visualstudio/ide/quick-actions?view=vs-2019)

波形曲线是波浪形下划线，它可以在键入时对代码中的错误或潜在问题发出警报。 这些可视线索使你能立即修复问题，而无需等待在生成期间或运行程序时发现错误。 如果将鼠标悬停在波形曲线上，将看到关于此错误的其他信息。 左边距中也可能会出现一个灯泡，提供修复此错误的“快速操作”建议。



 当光标位于可操作的代码行上时， 如果出现指示错误的红色波形曲线，且 Visual Studio 有针对该错误的可用修复方法，会显示一个错误灯泡 错误灯泡图标。

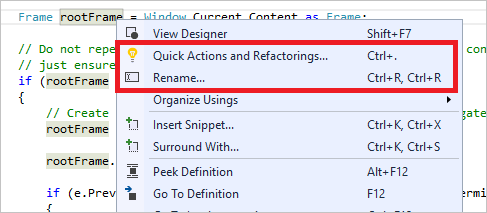
若要查看可能的修补程序，请选择灯泡旁边的下箭头或“显示可能的修补程序”链接。 此时会显示可用的“快速操作”列表。



### 2.重构(Refactor)

重构过程就是修改代码，以使其更易于维护、理解和扩展，但不会更改其行为。

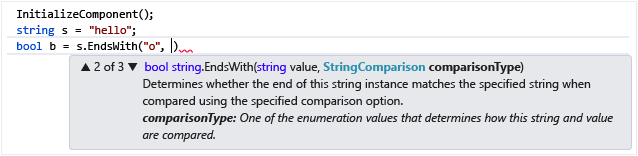
重构包括智能重命名变量、将一个或多个代码行提取到新方法中、更改方法参数的顺序等操作。



### 3.智能感知(IntelliSense)

IntelliSense 是代码完成辅助工具，它包含许多功能：成员列表、参数信息、快速信息和完成单词。 使用这些功能，可以详细了解使用的代码、跟踪键入的参数，还可通过轻敲几个按键来添加属性和方法调用。

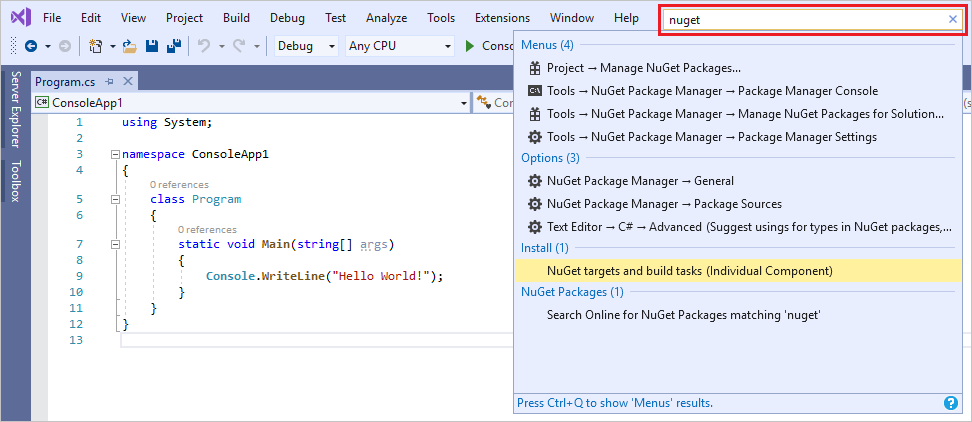
下图显示智能感知的快速信息功能，在输入代码的过程中快速的找到该函数的参数信息。



### 4.搜索框

Visual Studio 有时会因为有如此多的菜单、选项和属性而让人不知所措。 使用搜索框可以在 Visual Studio 中快速找到所需内容。开始键入要查找内容的名称时，Visual Studio 会列出结果，这些结果可以准确地将你导向目标位置。

**Tip**：按 Ctrl+Q 启动搜索框 。



# 安装Visual Studio

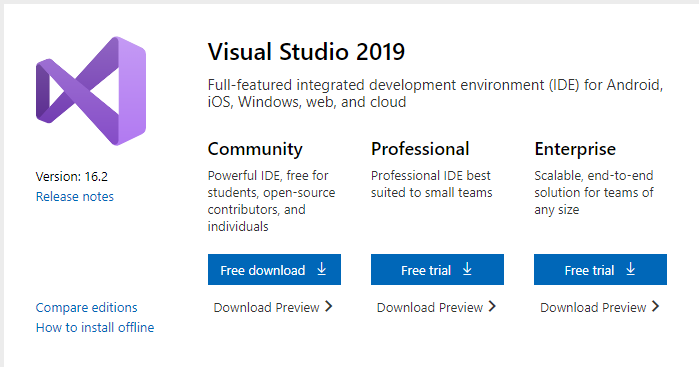
## 安装步骤

### 第 1 步：确保计算机支持 Visual Studio Community 2019

查看此处看操作系统的最低要求：  
<https://docs.microsoft.com/zh-cn/visualstudio/releases/2019/system-requirements>

### 第 2 步：下载 Visual Studio Community 2019

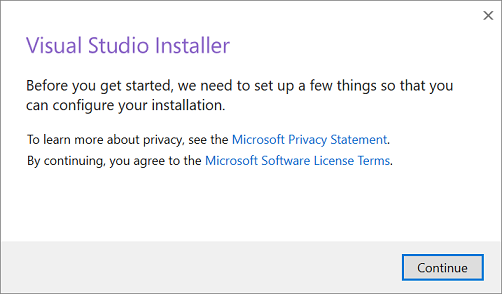
在此处下载：<https://visualstudio.microsoft.com/downloads/>



### 第 3 步：打开 Visual Studio Community 2019安装程序

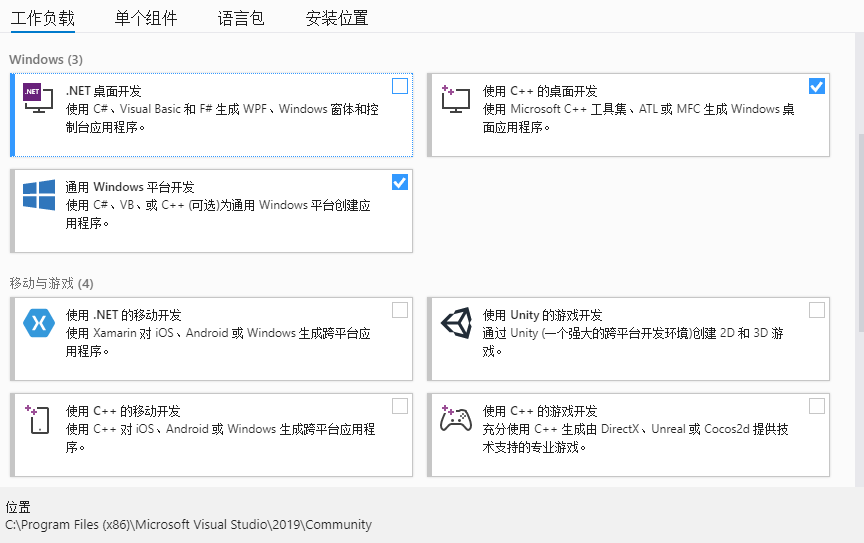
1.对于 Visual Studio Community，请运行**vs\_community.exe**

**2.** 确认 Microsoft [许可条款](https://visualstudio.microsoft.com/license-terms/)和 Microsoft [隐私声明](https://privacy.microsoft.com/privacystatement)。 选择“继续” 。

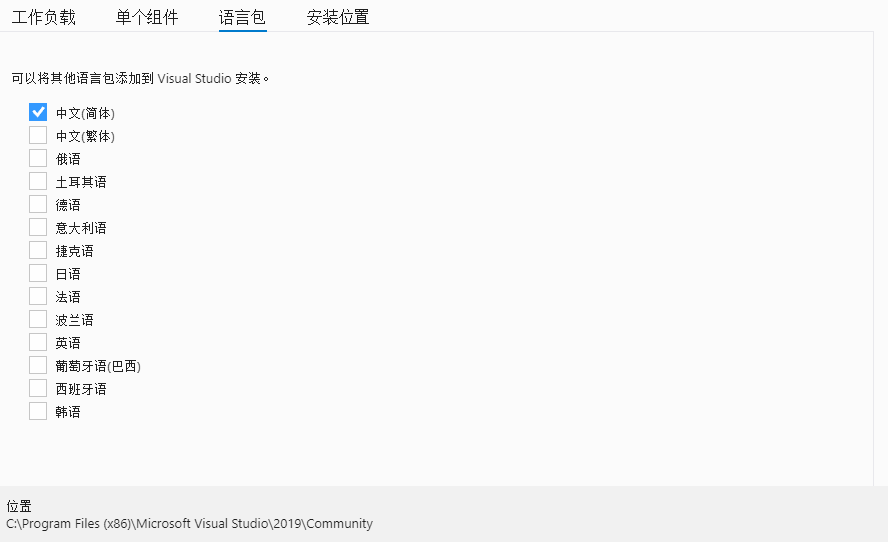


### 第 4 步：选择需要安装的工作负载

配置Visual Studio环境支持C++桌面开发：

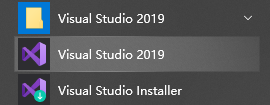


### 第5步：选择语言包：

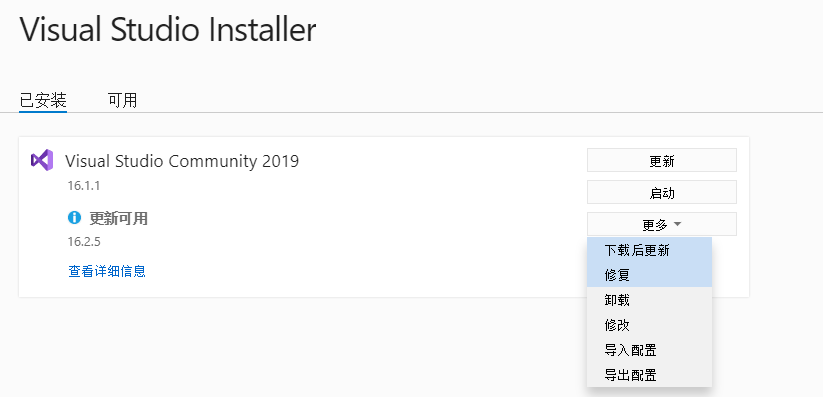


### 第6步：使用Visual Studio Installer

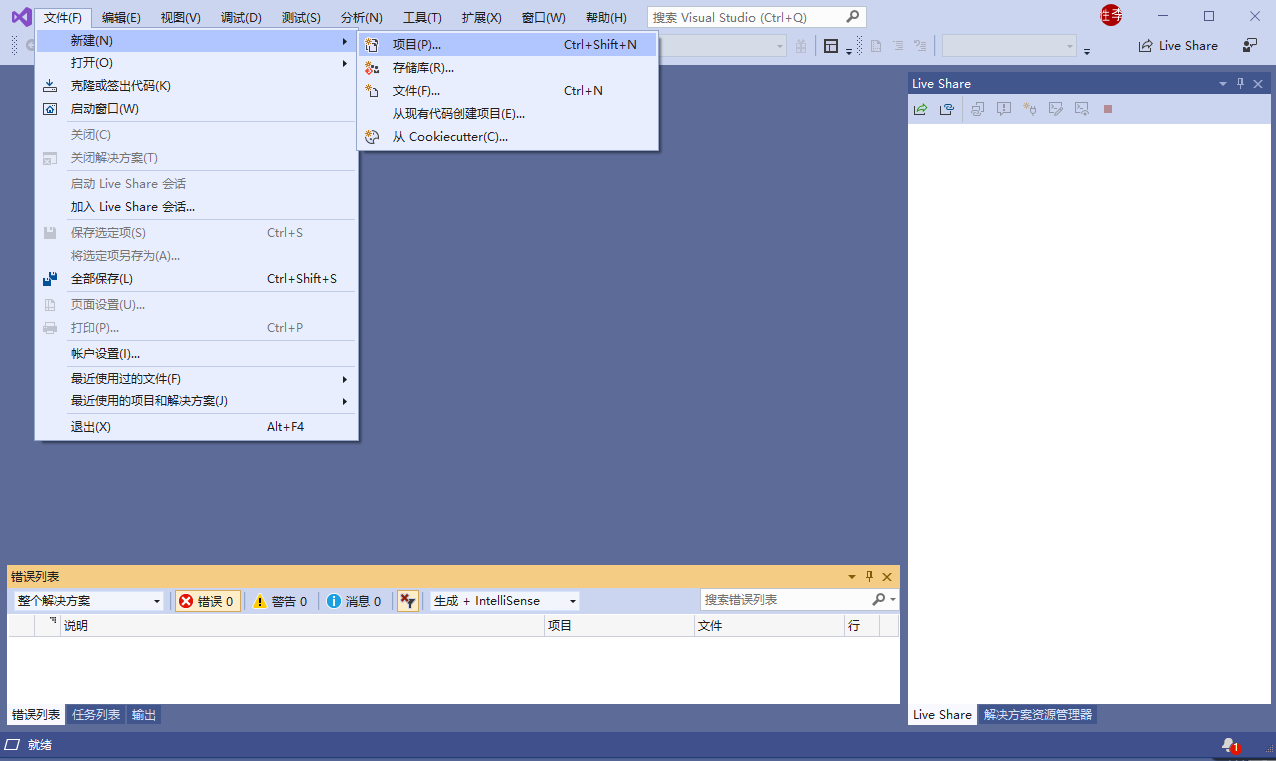
在开始菜单中找到Visual Studio Installer，单击运行之。



运行Visual Studio Installer可以卸载、修改Visual Studio 2019的配置。



### 第7步：运行Visual Studio



从新建->项目创建新的解决方案和项目，开始你的编程之旅吧！

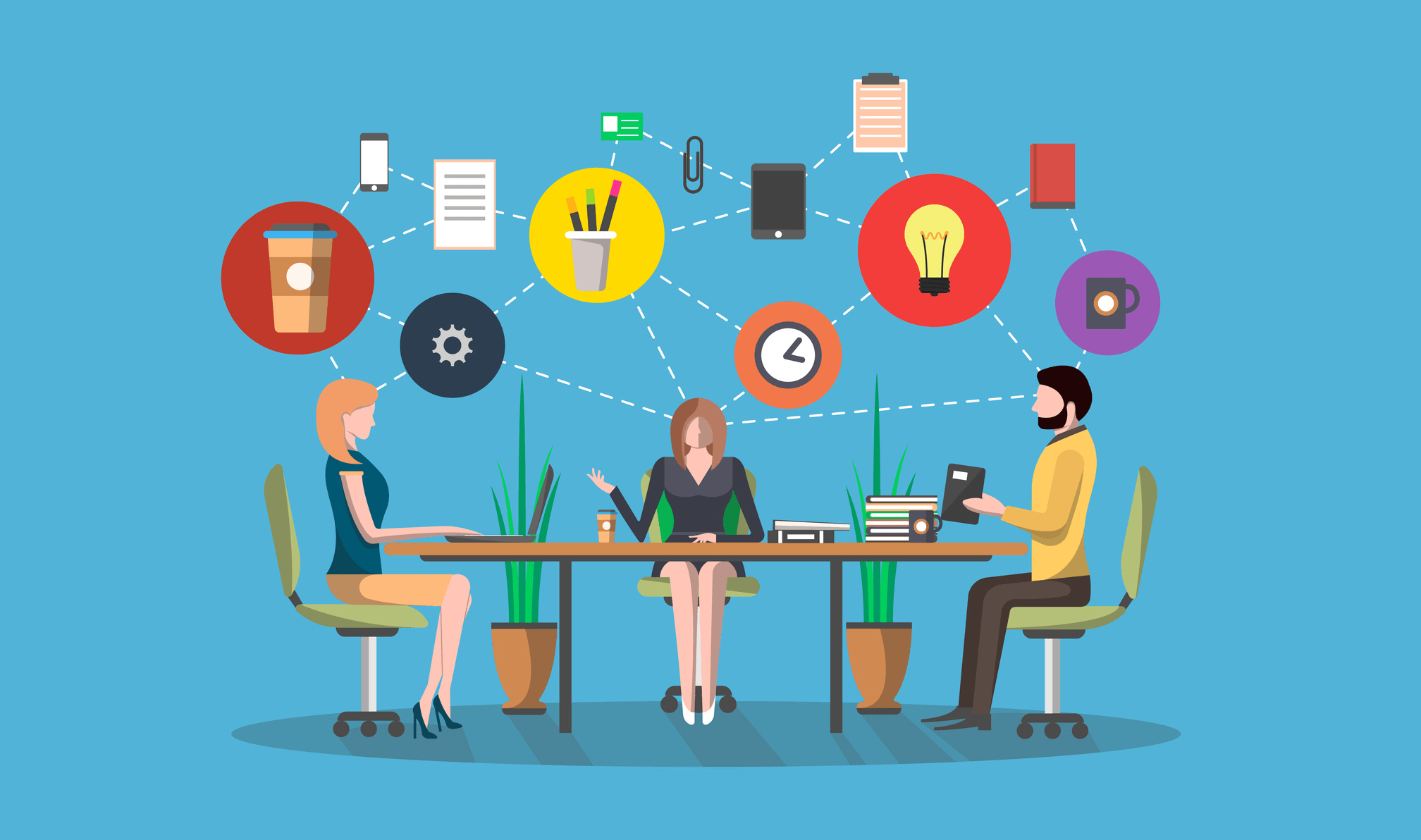
# Visual Studio 实用技巧[[2]](#footnote-2)

## 掌握项目与解决方案

与Visual Studio打交道的大部分时间都将花在项目或者解决方案内。虽然可以在Visual Studio中编辑孤立的、不和任何项目或解决方案相连的文件，但是使用的大多数文件都是存在于项目或者解决方案的环境中。

### 管理项目与解决方案

#### 什么是项目（Project）



Project: a piece of work that needs skill,effort,and careful planning,especially over a period of time

项目有许多用途，但是主要用于组织源代码，以便将其编译进库或者应用程序中。

#### 什么是解决方案（Solution）？



The method or process of solving a problem.

解决方案的定义：解决一个问题的方法或过程。

#### 解决方案(Solution)和项目(project)的关系



每个solution可以包含1个或者多个的projects.

解决方案把许多项目集合在一起，并且集中在单一的结构之下。比如，有一个由3个项目组成的业务解决方案，这3个部分可能包含数据访问层项目、业务逻辑层项目和表现层项目。解决方案将所有这些项目组合在一起，使得用户可以快速地在项目之间切换，并且管理它们之间的各种引用。

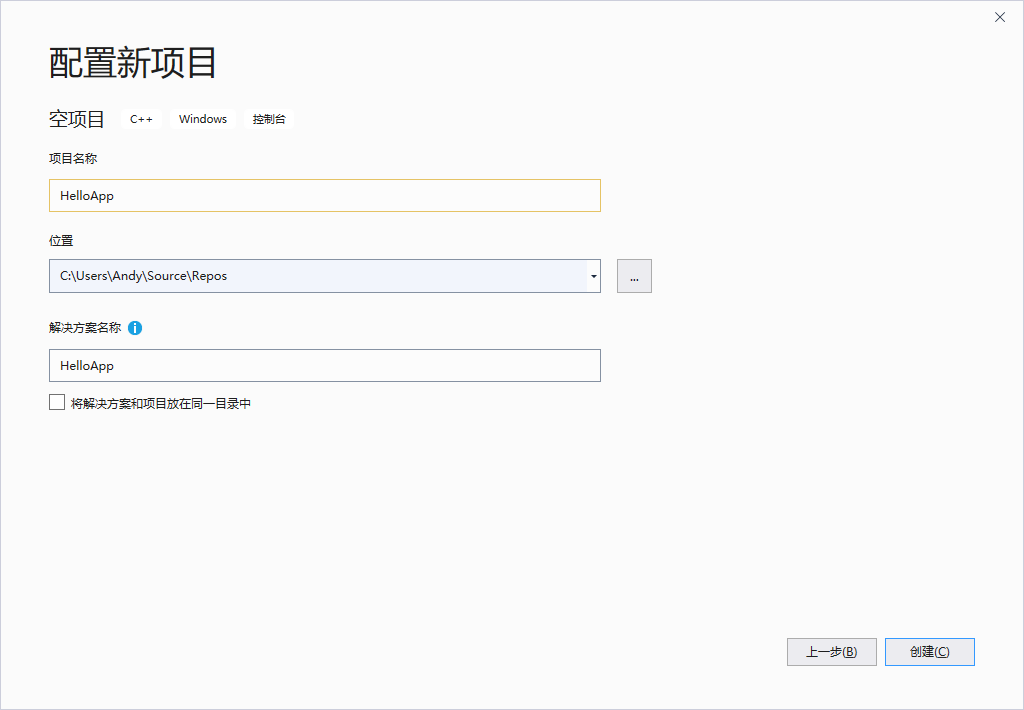
Note:解决方案是Visual Studio组织结构中的最高级别。

### 如何创建空白的新项目

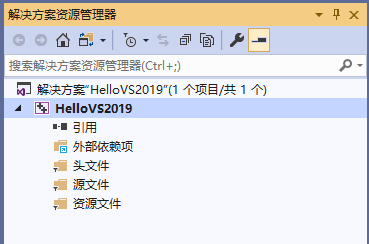


设置解决方案的名字为HelloVS2019，留意其存储位置。

本例子在： C:\Users\Andy\Source\Repos文件夹下。

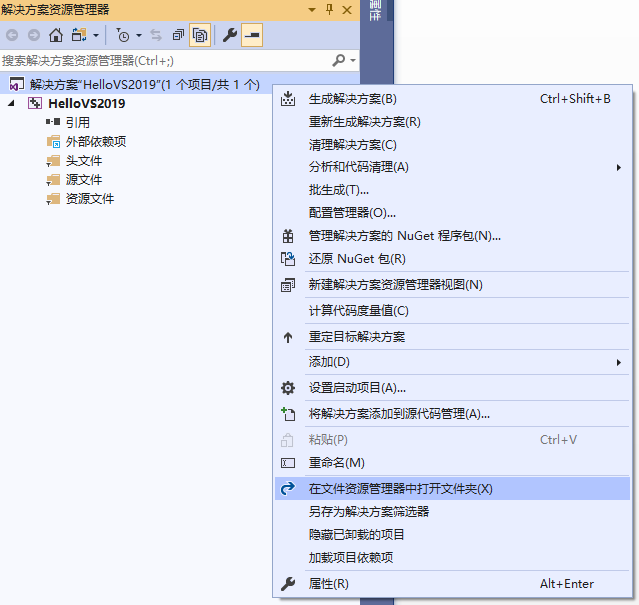


单击创建，生成的解决方案结构如下：



解决方案的名字为：HelloVS2019

项目的名字为：HelloVS2019



在文件资源管理器中打开文件夹，观察如下：



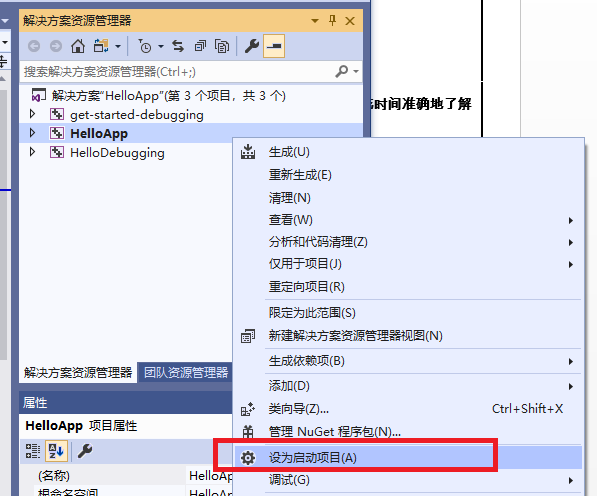
解决方案文件的扩展名是sln，项目在文件夹HelloVS2019之内。



项目的扩展名是vcxproj，其中关于这个项目的用户配置存储在HelloVS2019.user项目配置文件中。

### 设置解决方案的启动项目

启动项目(Startup Item)配置在调试器启动时Visual Studio应该执行的项目是哪个。

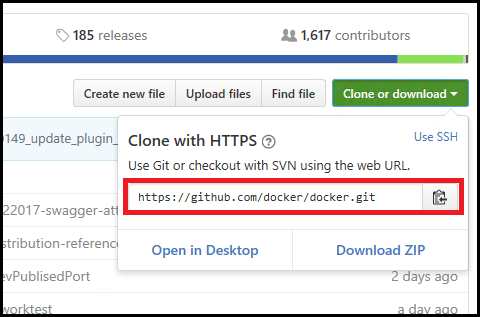


### 如何打开克隆的 GitHub 存储库中的代码[[3]](#footnote-3)

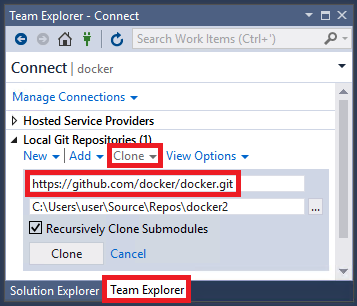
预备：以下示例演示如何克隆 GitHub 存储库，并在 Visual Studio 中打开其代码。

若要执行此过程，必须具有 GitHub 帐户，并且已经在系统上安装了适用于 Windows 的 Git。

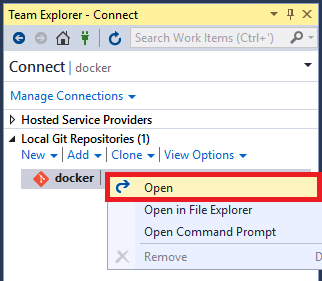
1. 转到要在 GitHub 上克隆的存储库。
2. 选择“克隆或下载”按钮，然后在下拉菜单中选择“复制到剪贴板”按钮，以便复制 GitHub 存储库的安全 URL。



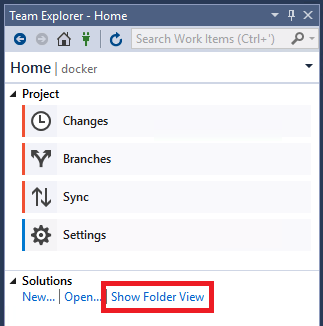
1. 在 Visual Studio 中，选择“团队资源管理器”选项卡，打开“团队资源管理器”。 如果未看到选项卡，则从“视图” > “团队资源管理器”中将其打开。
2. 在团队资源管理器的“本地 Git 存储库”部分中，选择“克隆”命令，然后将 GitHub 页的 URL 粘贴到文本框。



1. 选择“克隆”按钮，将项目的文件克隆到本地 Git 存储库。 此过程可能需要几分钟的时间，具体取决于存储库的大小。
2. 将存储库克隆到系统后，在“团队资源管理器”中，在新克隆存储库的上下文（右键单击）菜单上，选择“打开”命令。



1. 选择“显示文件夹视图”命令，查看“解决方案资源管理器”中的文件。



此时，可以浏览克隆存储库中的文件夹和文件，并在具有语法着色和其他功能的 Visual Studio 代码编辑器中查看和搜索代码。

## 二、掌握编辑器

尽管Visual Studio有许多功能，但它首先是个代码编辑器。编辑器可能是我们在Visual Studio上工作花时间最多的地方，所以我们需要充分了解编辑器的各种功能，并且把其中一些基本的功能发挥到极致。与普通的文本编辑器相比，Visual Studio的编辑器提供了诸如语法着色，智能感知，代码格式化等诸多功能。

Visual Studio的编辑器设计了丰富的键盘快捷键，在编辑代码的过程中，充分使用快捷键可以节约我们的手离开键盘去移动鼠标的时间。Visual Studio的设计目标之一是开发人员可以完全不适用鼠标完成所有的开发工作，这意味着这个基本上每一个菜单每一个功能都有相应的键盘快捷键。善用键盘快捷键，可以极大的提高开发效率。

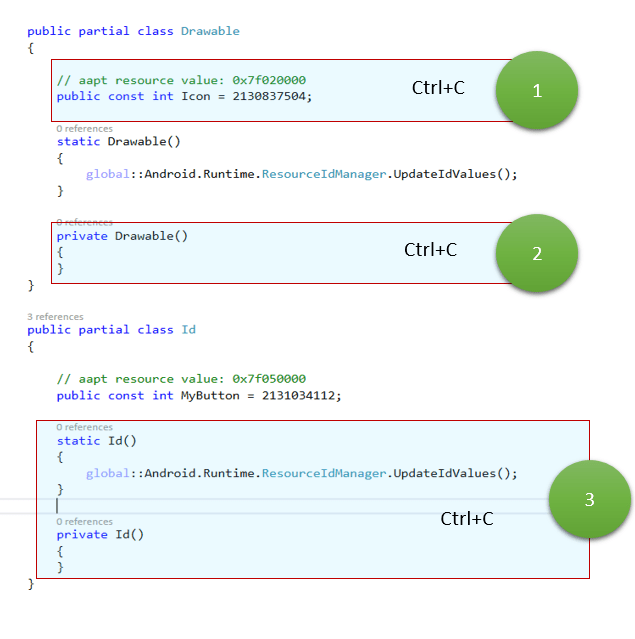
### 掌握剪切板

在编辑代码的过程中，最常见的功能就是使用剪切板，执行复制Ctrl+C和粘贴Ctrl+V操作。

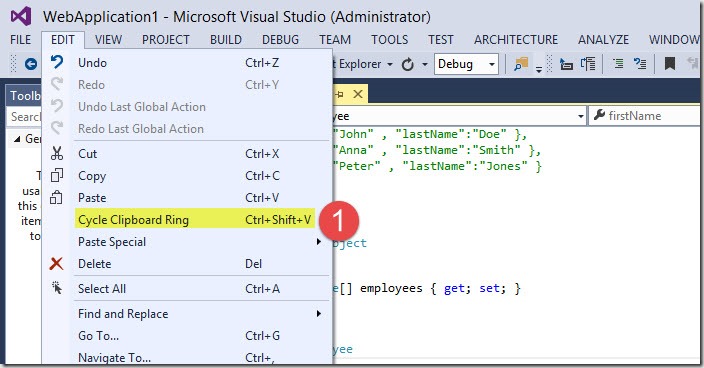
在应用程序中复制、粘贴文本时，通常一次只能复制粘贴一个项目。如果想要复制粘贴两条不同的语句，就要先复制第一条，粘贴第一条，然后复制第二条，再粘贴第二条。如果有10个语句需要复制，并且将这10个语句粘贴在文档的不同地方，这个过程就会相当麻烦。VS中的剪切板循环，改进了这个繁琐的复制粘贴过程。

#### 剪切板循环（Cycle Clipboard Ring）

在VS中可以连续Ctrl+C多段不同的文本，VS会将每次Ctrl+C的文本放在剪切板循环中。



然后按Ctrl+Shift+V显示剪切板的内容，选择哪一项要粘贴在目标位置中。



#### 基于行的剪切与粘贴

VS做了一个非常简单的假设，如果当前没有选择任何想要剪切、复制或删除的文本，那么任何编辑操作将默认运用于当前行。

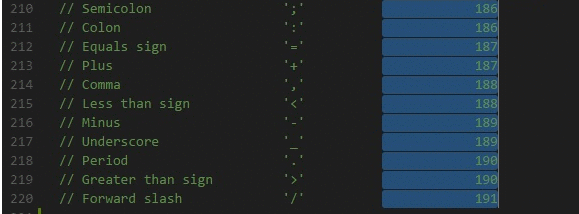
因此，如果想要把一行复制粘贴到另外的位置，可以不需要选择任何文本。只需要把光标停留在那行之中，按Ctrl+C，然后移动到目标位置按Ctrl+V。既可以复制粘贴。

同样的，如果想要把整行剪切，只需要把光标停留在那行，按Ctrl+X即可。

因此，使用Ctrl+C，Ctrl+X就可以快速的拷贝、剪切当前行。

#### 快选择Column (box) selection

按住Alt+Shift后移动键盘上的方向键或者移动鼠标。

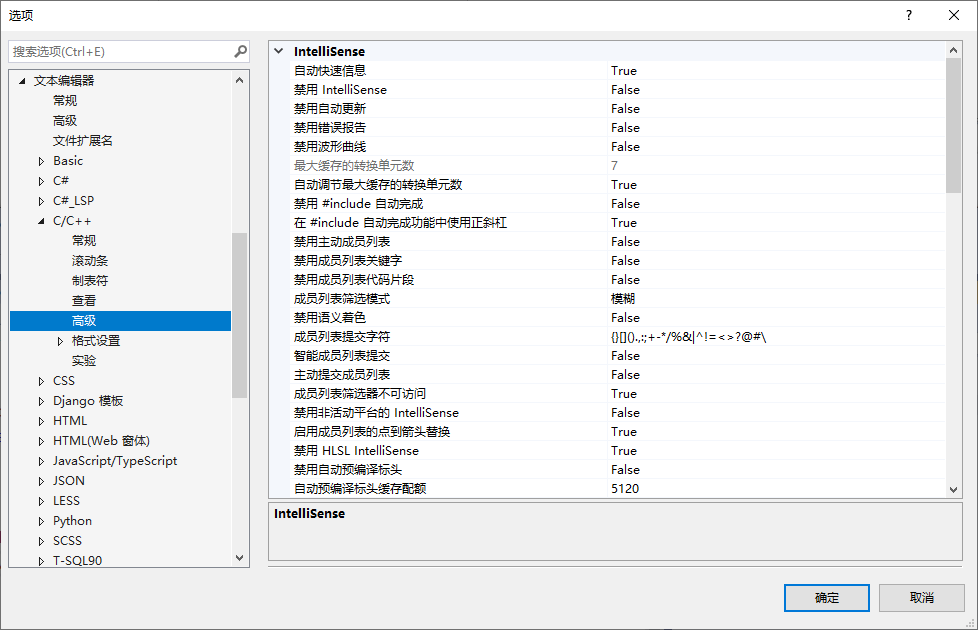


VS会在每行的末尾会添加一个独立的光标，借着可以同时编辑多行。

### 掌握智能感知Intellisence[[4]](#footnote-4)

IntelliSense 是使编码更方便的一组功能的名称。这些功能是程序员的大爱！

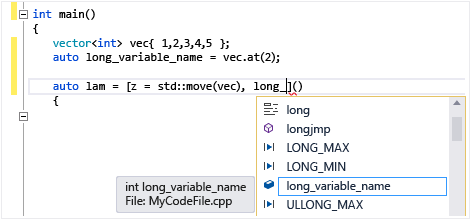
几乎所有的 IntelliSense 功能都可以在“选项” 对话框中的“文本编辑器” > “C/C++” > “高级” 下启用或禁用。 可从菜单栏上的“工具” 菜单中访问“选项” 对话框。



#### Intellisence的快捷键如下

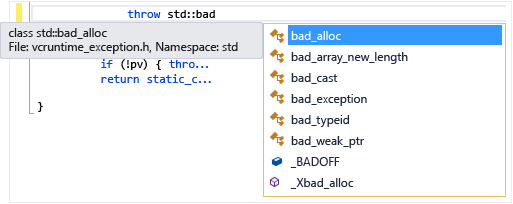


* 完成单词 Ctrl+Space



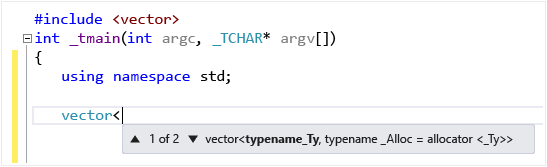
如上图，在输入long\_之后，按Ctrl+Space ,Intellisence列出供补齐的变量名，选择所要的单词按TAB即可补齐。

* **列出成员的快捷键是** Ctrl+J



如图，在输入throw std::bad之后，Intellisence列出可供选择的成员列表。

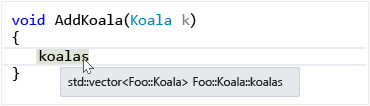
* **参数帮助**



如上图，当你在类模板变量声明上键入函数调用的左大括号或尖括号时，该编辑器将显示具有函数或构造函数的每个重载的参数类型的小窗口。

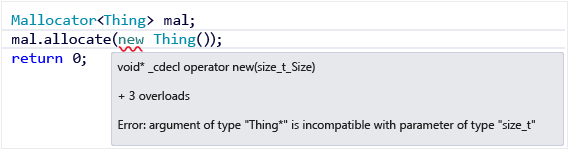
* **快速信息**

将鼠标光标悬停在变量上时，将在内联出现一个小窗口，显示类型信息和在其中定义该类型的标头。 将鼠标悬停在函数调用上，以查看该函数的签名。



* **错误波形曲线**

程序元素（变量、关键字、大括号、类型名称等）下的波形曲线提醒你注意代码中的错误或潜在错误。当你编写前向声明时，会出现绿色波形曲线，提醒你仍然需要编写实现。红色波形曲线指示处于活动状态的代码中存在需要处理的编译器错误或警告。

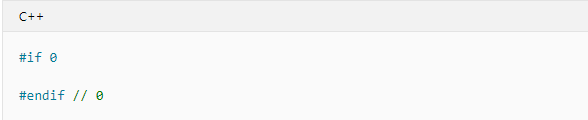


* **代码片段[[5]](#footnote-5)**

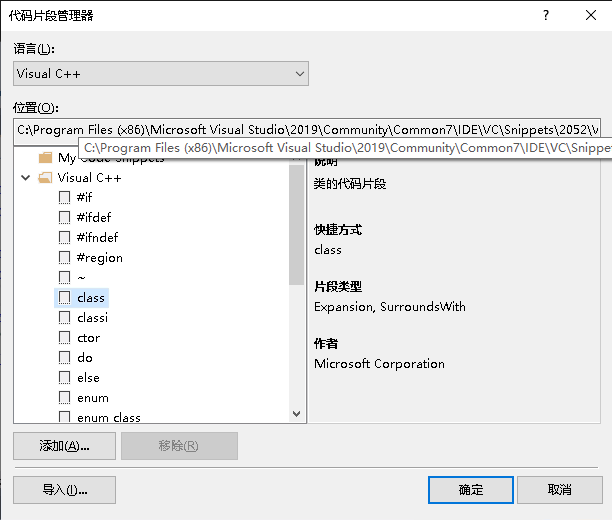
在键盘上按“Ctrl+K+X 然后选择要插入的代码片段：

将出现一个以“#if” 开头的选择列表。

选择“#if” 后，显示添加到文件的以下代码：

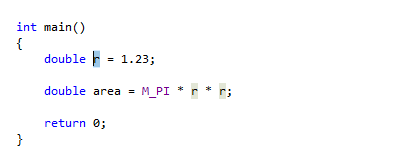


使用代码片段管理器可以自定义常用供插入的代码片段



* 重命名[[6]](#footnote-6)

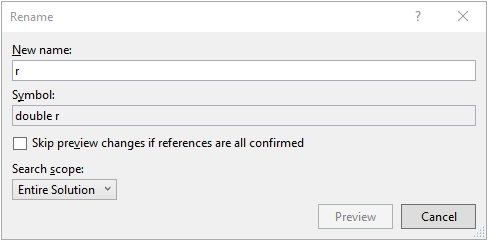
1.突出显示要重命名的项，或将文本光标置于其中：



2. 鼠标

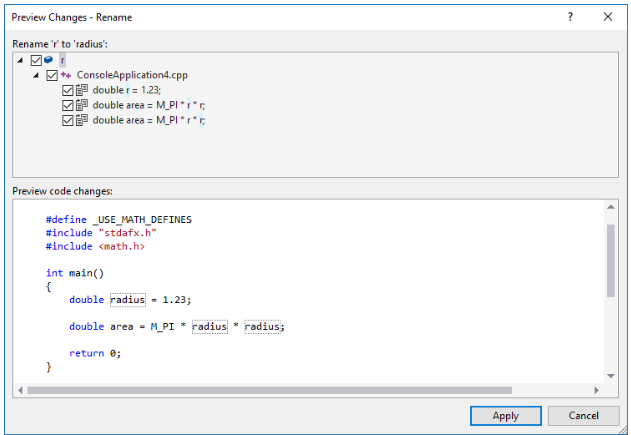
右键单击代码并选择“重命名”。

3.在弹出的“重命名”窗口中，输入所选项的新名称，并单击“预览”按钮。



4.当出现“预览更改 - 重命名”窗口时，请确保正确进行所请求的更改。

使用窗口上半部分的复选框，启用或禁用任何项的重命名。



5. 单击“应用”按钮，项将在源代码中重命名。

## 三、调试[[7]](#footnote-7)

毫无疑问，软件开发人员编写的代码并不总是按照预期行事。 有时它执行一些完全不同的操作！ 发生这种情况时，下一个任务是找出原因。尽管我们可能会一直盯着代码看几个小时，但使用调试工具或调试程序会更轻松且更高效。

遗憾的是，调试程序无法神奇地揭示代码中的所有问题或“bug”。 调试意味着在 Visual Studio 等调试工具中逐步运行代码，以找到导致编程错误的确切位置。 那么就可以了解代码中所需的更正，并且调试工具通常允许进行临时更改，以便可以继续运行该程序。

有效地使用调试程序也是一项需要时间和实践来学习的技能，但从根本上来说这是每个软件开发人员的一项基本任务。

### 调试的核心原则

#### 第一步：自我拷问

因此，开始调试之前，要问自己的问题：

你的期待：代码可执行哪些操作？

事实如何：实际上发生了什么？

不断反复的问自己这些问题，层层追问，然后运用调试工作找到使事实与期待不同的代码。

调试工具有助于找到代码中发生异常的确切位置并且帮助调查可能的修复方法。

#### 第二步：检查假设

在调查 bug 或错误之前，请考虑你期望获得某个结果的假设。 即使正在查看调试程序中问题的成因，隐藏或未知的假设也可能妨碍识别问题。 可能有一长串可能的假设！ 以下是要询问的一些问题，以质疑你的假设。

* 是否使用正确的 API（即正确的对象、函数、方法或属性）？
* 是否正确地使用了 API？
* 代码是否包含任何拼写错误？
* 是否对代码进行了更改，导致出现新问题？
* 是否期望对象或变量包含与实际情况不同的特定值（或某种类型的值）？
* 是否知道代码的含义？ （调试其他人的代码通常更加困难。 如果不是你的代码，则可能需要花时间准确地了解代码的作用，然后才能有效地进行调试。）

通过质疑假设，可以减少在代码中查找问题所需的时间。 此外可缩短解决问题所需的时间。

#### 第三步：单步调试代码

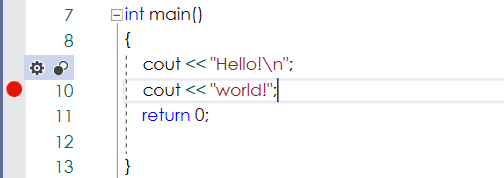
在 “调试模式”中启动程序，这意味着调试程序会主动监视程序运行时发生的所有事情，并且允许在任何时候暂停应用以检查其状态，并且可以启用逐行单步调试代码以查看运行过程中发生的每个细节。

在 VS 中，通过快捷键F5来进入调试模式，或者点击调试工具栏的按钮。

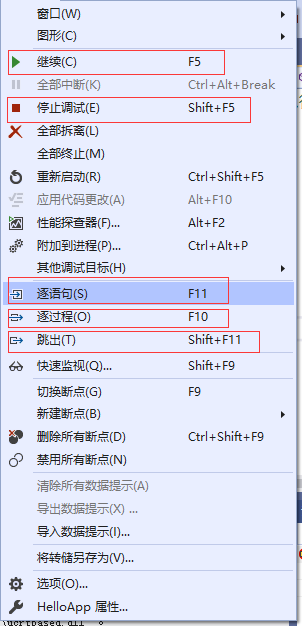


  断点是调试的最基本和最重要的功能。 断点指示 Visual Studio 应在何处暂停正在运行的代码，以查看变量的值或内存的行为，或代码运行的顺序。

在 Visual Studio 中，可通过单击代码行旁边的左边距来快速设置断点 或将光标置于行并按 F9。



调试菜单如下：



包含停止调试、逐语句执行、逐过程执行、跳出等功能，其对应的工具栏如下图：



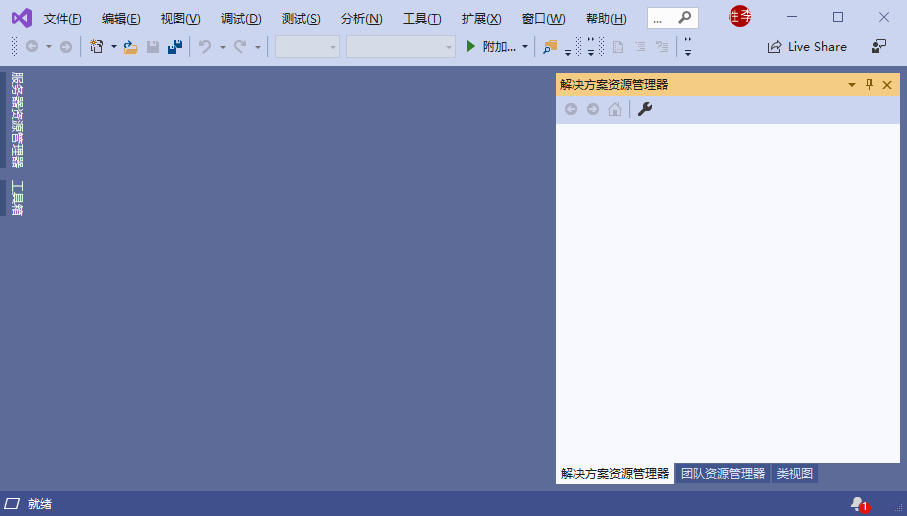
# Visual Studio 中的 C++ 入门

## 第一个C++程序HelloApp[[8]](#footnote-8)

知识点：解决方案，添加项目，输出Hello，生成exe，设置断点，清理工程，发布生成release版的exe

### 创建控制台应用

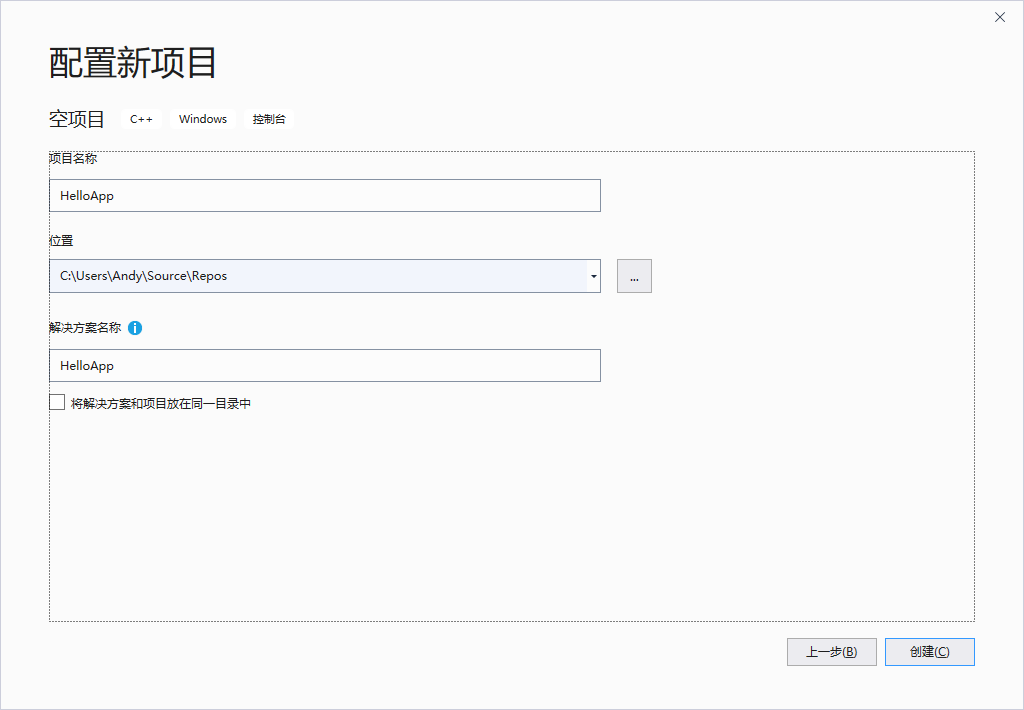
1. 启动VS



1. 创建新项目 Ctrl+Shift+N



选择空项目(C++,Window,控制台)，单击下一步。



设置项目名称和解决方案名称都为HelloApp,单击创建。

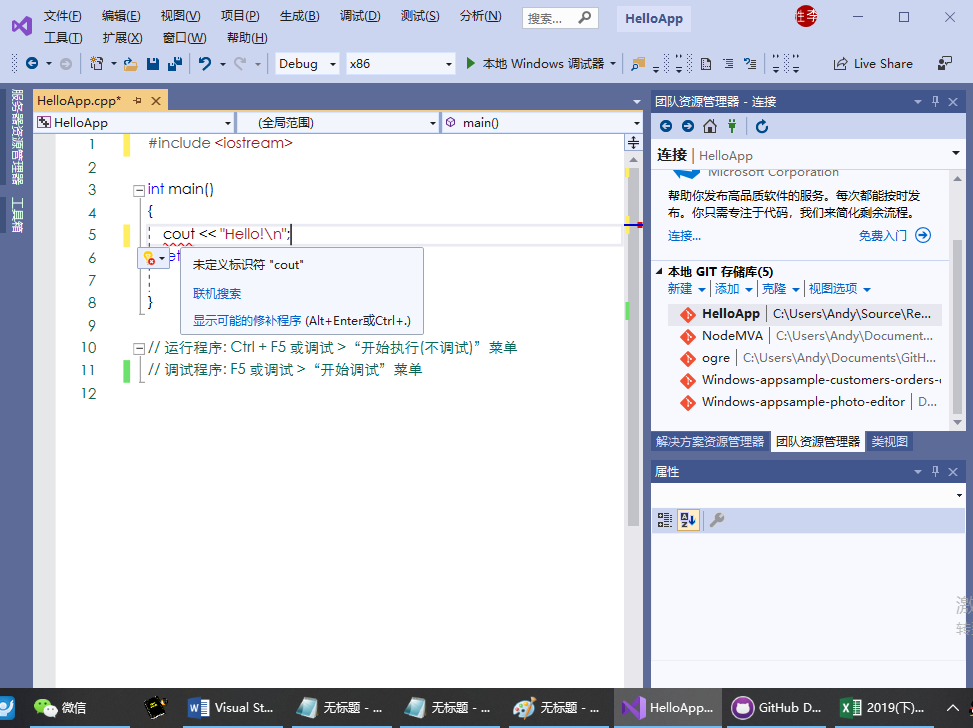
### 向应用添加代码

**输入**#include <iostream>

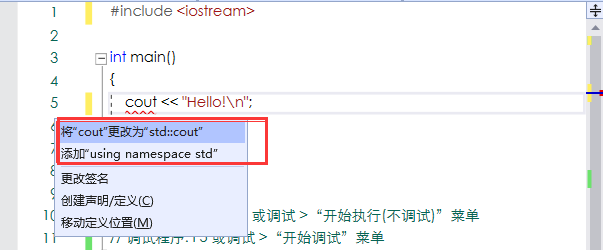
输入cout<<”hello!/n”;



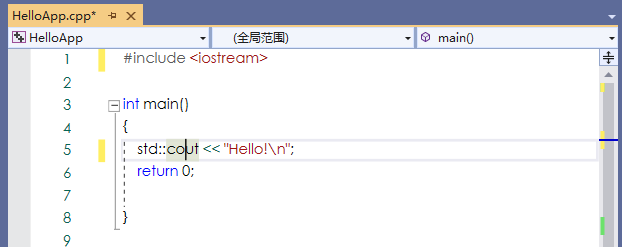
红色的波浪线显示在 cout下面。 如果将指针悬停在其上方，则会显示错误消息。



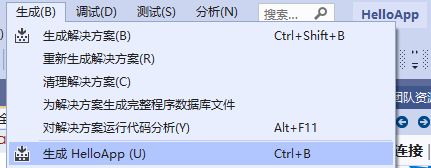
按Alt+Enter



在显示出的列表中选择第一项，回车，代码自动修正如下：

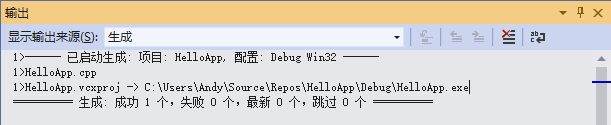


### 生成应用



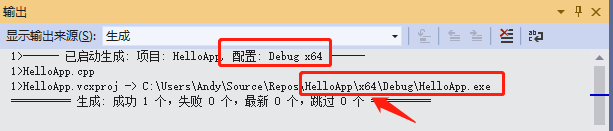
单击菜单生成🡪生成解决方案，或者使用快捷键 Ctrl+Shift+B

可以生成exe应用程序。打开输出窗口可以看到生成HelloApp.exe的位置：



这个exe是按照这个输出类型生成32位程序。

如果生成设置更改为将会生成64位的应用程序:



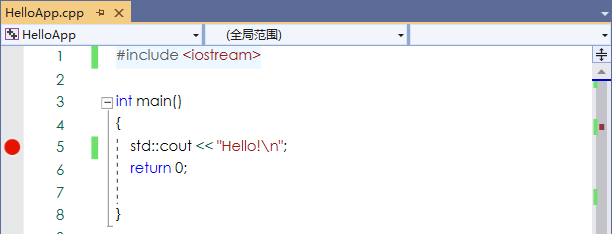
### 调试和测试应用

可调试 HelloApp，查看控制台窗口中是否显示单词“Hello”。

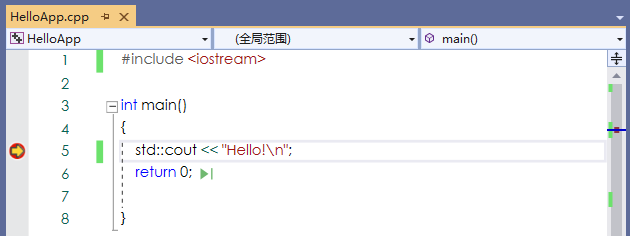
若要调试应用程序，可以按F5以调试模式运行程序

若要设置断点，可在选定的代码行上按F9**，设置断点**

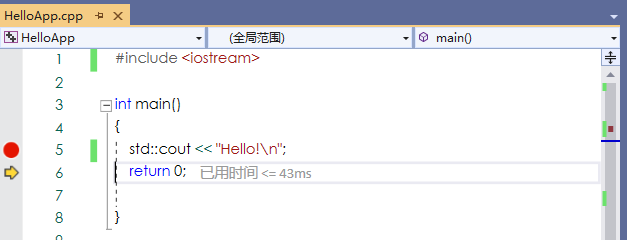
如图，设置断点如下：



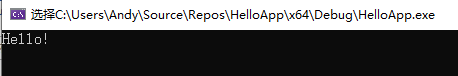
按F5运行到断点的位置：



接着按F10逐过程运行代码。



查看控制台输出窗口：



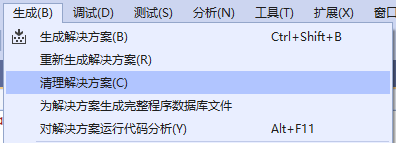
程序调试成功，控制台窗口出现并显示单词 **Hello。**

若要停止调试，请按 Shift+F5 。

### 生成应用程序的发布版本

确认一切就绪后，可以准备该应用程序的发布版本。 发布版本省去了调试信息，且使用编译器优化选项来创建更小、更快的代码。

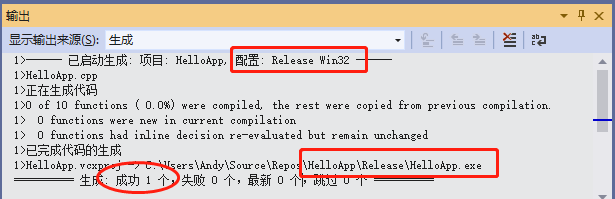
**第一步：要清理解决方案文件并生成发布版本**



**第二步：将 HelloApp 的解决方案配置从“调试(debug)”更改为“发布(release)”**



第三步：生成解决方案。 在菜单栏上，依次选择“生成”>“生成解决方案” 。



**第四步：打开资源管理器并且查看运行HelloApp.exe**



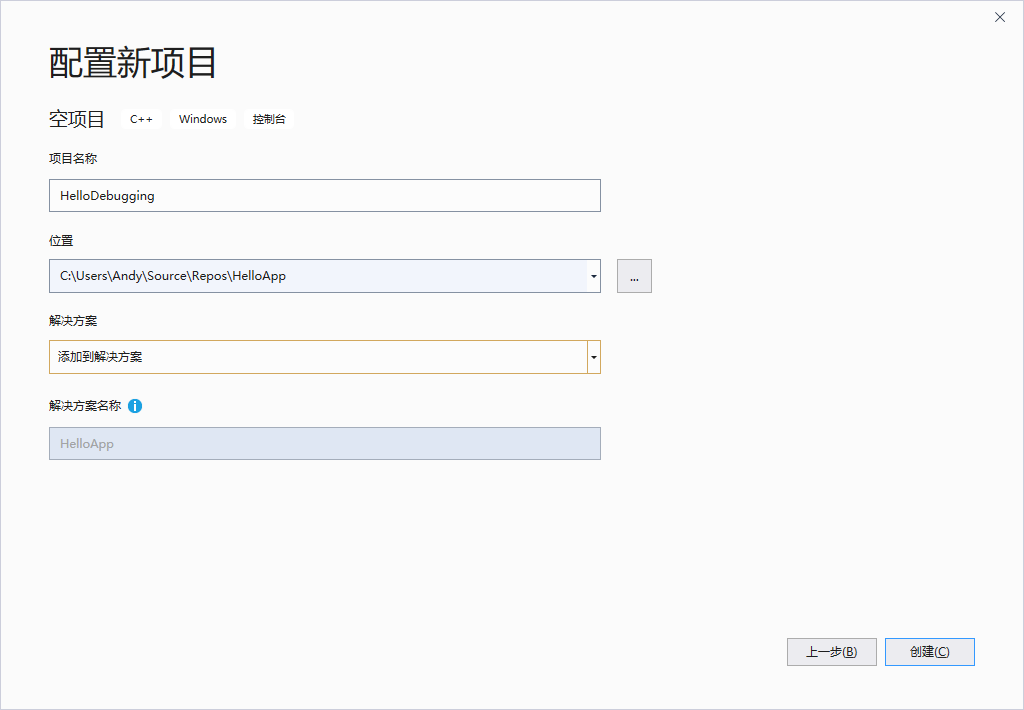
**找到exe的位置如下：**



## 调试C++程序[[9]](#footnote-9)

知识点：解决方案中添加新项目，添加C++文件，设置断点，变量监视器，单步调试

### 在HelloApp.sln之中，添加新的空白项目

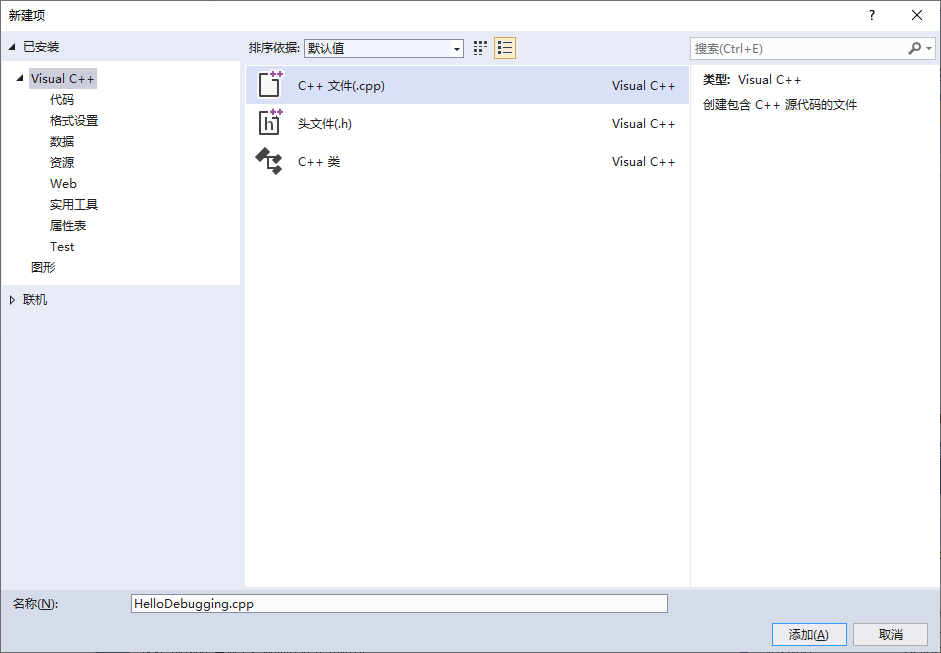


### 按Ctrl+Q,搜索C++文件模板，并且添加到项目中

类型:C++文件(cpp)

文件名:HelloDebugging

设置如下：



### 添加代码

把HelloDebugging.cpp的代码更新如下：

#include <list>

#include <iostream>

using namespace std;

void doWork()

{

list <int> c1;

c1.push\_back(10);

c1.push\_back(20);

const list <int> c2 = c1;

const int &i = c2.front();

const int &j = c2.front();

cout << "The first element is " << i << endl;

cout << "The second element is " << j << endl;

}

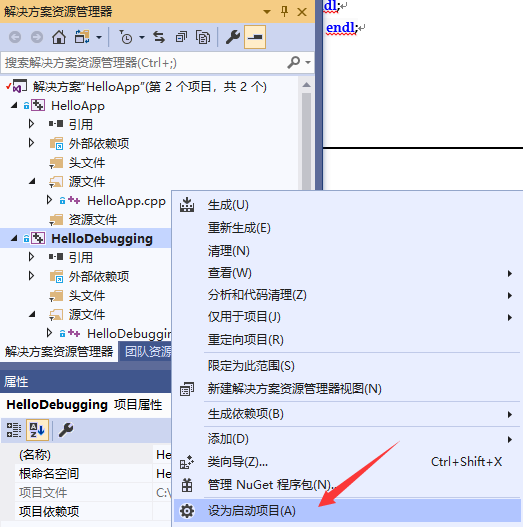
int main()

{

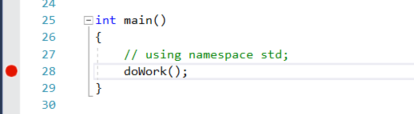
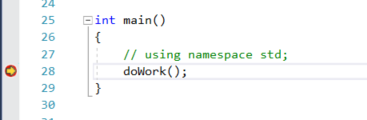
doWork();

}

### 设置HelloDebugging为启动项目



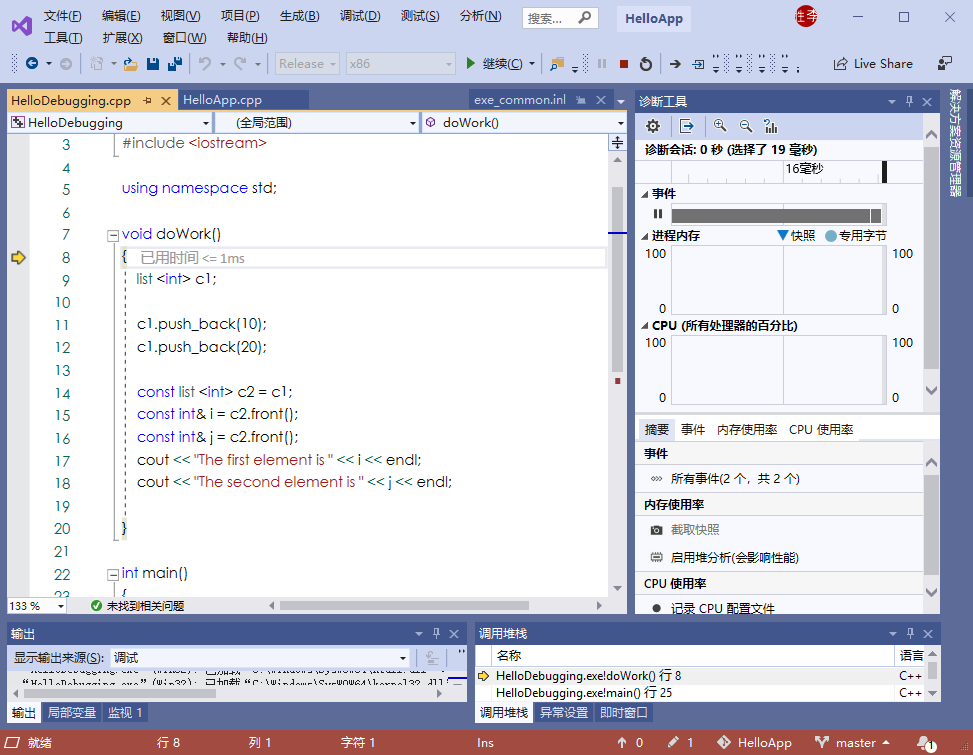
### 设置断点并且运行到断点

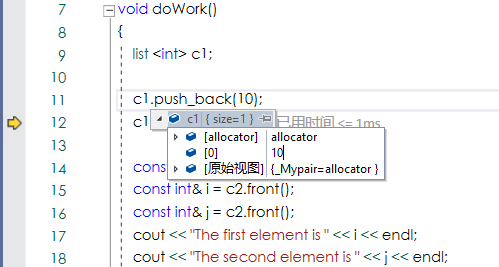
**设置断点：**F9 **按**F5**执行到此断点。**

### 单步调试

按F11进入doWork()过程内逐语句调试。



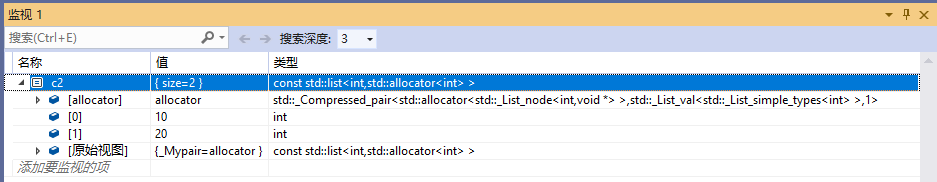
按F10逐过程执行到第12行代码，鼠标悬停在C1变量上，可以查看该变量的详细信息。该列表此时存储整数10，详细信息如下：



### 监视窗口



添加对C2变量的监视结果如下：

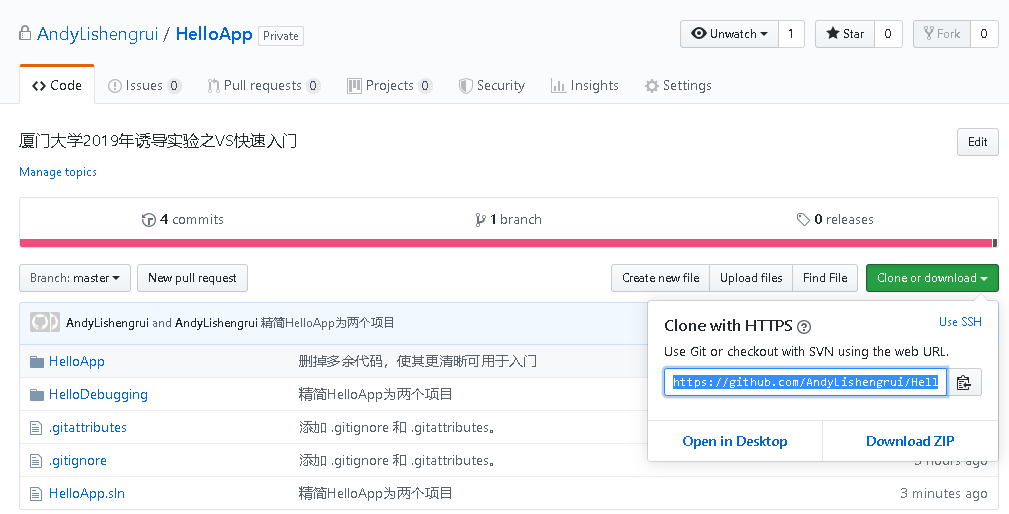


使用监视窗口可以查看变量在内存中的值，帮助我们调试代码，了解程序的运行状态。

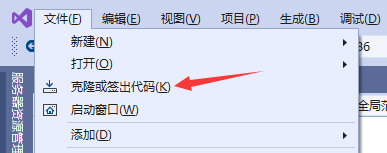
## 在Visual Studio中添加Github项目

1. Clone本教程的示例代码HelloApp.sln

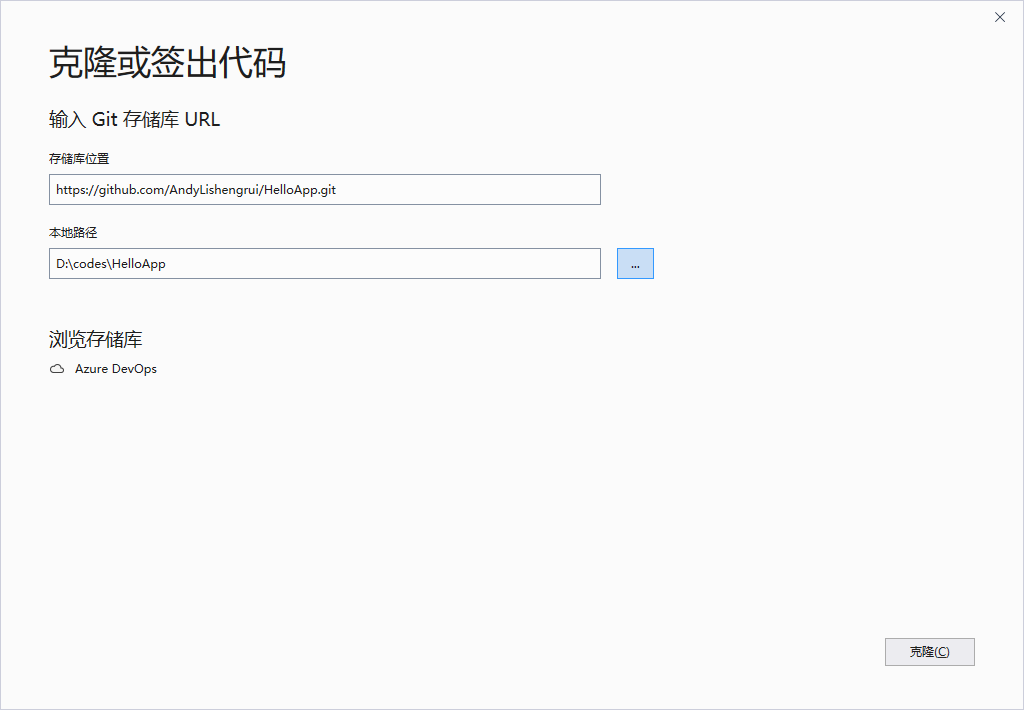
<https://github.com/AndyLishengrui/HelloApp.git>



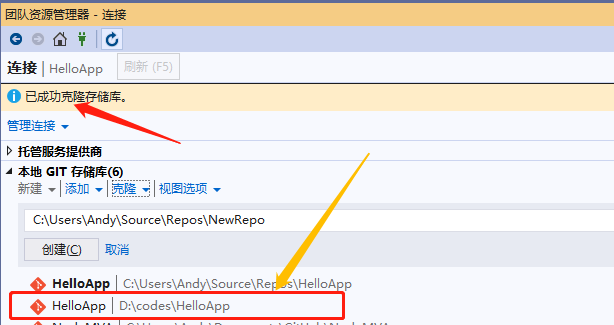
1. Clone代码



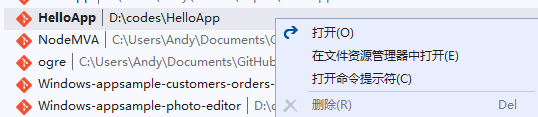
指定一个空文件夹clone该工程。



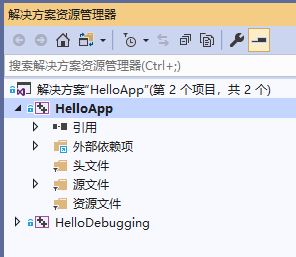
在团队资源管理器里面管理连接，查看clone是否成功，若成功提示如下：



选择要打开的GIT存储库



在解决方案管理器中打开该工程：



恭喜你，这就是本学期的Visual Studio快速入门的全部内容了，盼望未来的你更多熟悉这个开发工具，在程序设计师的人生路上带着喜乐和Hack的精神前行！

1. 参考: <https://docs.microsoft.com/zh-cn/visualstudio/get-started/visual-studio-ide?view=vs-2019> [↑](#footnote-ref-1)
2. 《VISUAL STUDIO HACKS》 作者: 艾弗里 出版社: 清华大学出版社 [↑](#footnote-ref-2)
3. 来源:<https://docs.microsoft.com/zh-cn/visualstudio/ide/develop-code-in-visual-studio-without-projects-or-solutions?view=vs-2019> [↑](#footnote-ref-3)
4. Visual C++ IntelliSense 功能 <https://docs.microsoft.com/zh-cn/visualstudio/ide/visual-cpp-intellisense?view=vs-2019> [↑](#footnote-ref-4)
5. Visual C++ 代码片段 <https://docs.microsoft.com/zh-cn/visualstudio/ide/visual-cpp-code-snippets?view=vs-2019> [↑](#footnote-ref-5)
6. 重构C++代码之重命名 <https://docs.microsoft.com/zh-cn/cpp/ide/refactoring/rename?view=vs-2019> [↑](#footnote-ref-6)
7. 初步了解 Visual Studio 调试器<https://docs.microsoft.com/zh-cn/visualstudio/debugger/debugger-feature-tour?view=vs-2019> [↑](#footnote-ref-7)
8. HelloApp: https://docs.microsoft.com/zh-cn/cpp/get-started/tutorial-console-cpp?view=vs-2019 [↑](#footnote-ref-8)
9. 调试C++程序https://docs.microsoft.com/zh-cn/visualstudio/debugger/quickstart-debug-with-cplusplus?view=vs-2019 [↑](#footnote-ref-9)