

目录

1、支持的平台软件版本.....	2
2、学术版与企业版功能对比.....	2
3、平台软件安装注意事项.....	3
3.1、Visual Studio 安装注意事项.....	3
3.2、Intel Visual Fortran 安装注意事项.....	6
3.3、Matlab 安装注意事项.....	7
4、UDF 编译调试功能基本使用步骤.....	8
5、调用 CoolProp 函数基本使用步骤.....	15
6、调用 Intel Fortran 功能基本使用步骤.....	16
7、调用 Matlab 功能基本使用步骤.....	20
8、设置第三方库目录（仅企业版）.....	24
9、拓展函数库.....	25
9.1、开启拓展函数库.....	25
9.2、拓展函数详解.....	25
9.3、学术版拓展函数实例.....	27
10、小贴士.....	28
11、如何注册.....	29

1、支持的平台软件版本

表 1、支持的平台软件版本

平台软件版本	是否支持
WinXP~Win11 (x86/x64)	✓
Fluent 6.3~2023R2(x86/x64)	✓
Visual Studio 2010~2019 (Express 版除外)	✓
Visual Studio 2022 或更高	✗
Intel Visual Fortran 2011~2018 (可选安装)	✓
Matlab2014a~2021b (可选安装)	✓

*推荐使用 Win10 + Visual Studio 2015 中文社区版+ Fluent 17 或更高

2、学术版与企业版功能对比

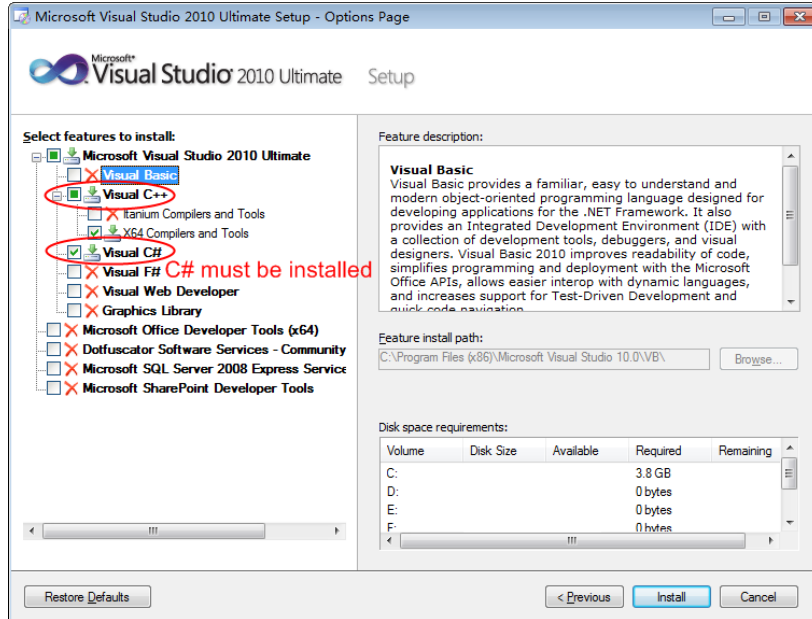
表 2、学术版与企业版功能对比

功能	学术版		企业版	
	试用版	注册版	试用版	注册版
编译调试（串行，单精度）	✓ 最多 1 个宏 Fluent2022R2 或更高无串行试用	✓ 宏数不限	✓ 最多 1 个宏 Fluent2022R2 或更高无串行试用	✓ 宏数不限
编译调试（串行，双精度）	✗	✓	✗	✓
编译调试（并行，单/双精度）	✗ Fluent2022R2 或更高单核并行试用	✓	✗ Fluent2022R2 或更高单核并行试用	✓
调用 C++/Win32 API/MFC 函数	✓	✓	✓	✓
UDF 中根据区域名字获取 ID 号	✓	✓	✓	✓
UDF 语句中断迭代	✓ 仅限串行单精度	✓ 不限	✓ 仅限串行单精度	✓ 不限
调用第三方 LIB 库	✓	✓	✓	✓
设置第三方函数库目录	✗	✗	✓ 最多 1 个目录	✓ 目录数不限
Fluent 中加入用户菜单	✗	✗	✓ 最多 2 个子菜单	✓ 子菜单数目不限
UDF 中驱动 fluent 迭代	✗	✗	✓ 最多 1 次迭代	✓ 迭代次数不限
UDF 中调用 Scheme/TUI 命令	✗	✗	✗	✓
由 Workbench 启动	✗	✓	✗	✓
调用 CoolProp 函数(可选采购)	✓ 最多 1 个函数	✓ 无限制	✓ 最多 1 个函数	✓ 无限制
调用 Fortran 子过程(可选采购)	✓ 最多 1 个子过程	✓ 无限制	✓ 最多 1 个子过程	✓ 无限制
调用 Matlab 子函数(可选采购)	✓ 最多 1 个子函数 输入参数不允许矩阵 不允许动态链接模式	✓ 无限制	✓ 最多 1 个子函数 输入参数不允许矩阵 不允许动态链接模式	✓ 无限制

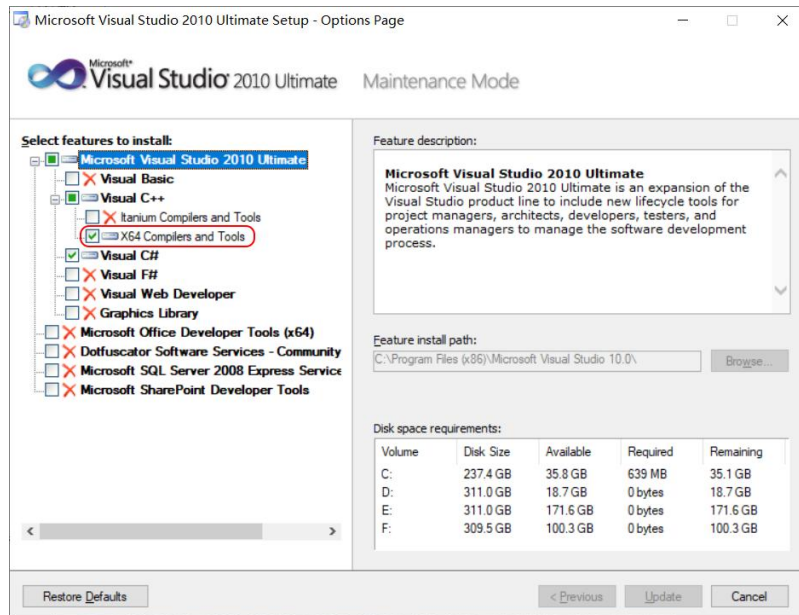
3、平台软件安装注意事项

3.1、Visual Studio 安装注意事项

1. 建议 **Visual C#** 与 **Visual C++** 同时安装。尽管对于某些 **Visual Studio** 版本这并不是必须的，但对于 **VS2010** 版本，**Visual C#** 是必须安装的，否则软件启动 **Visual Studio** 的时候会出错。

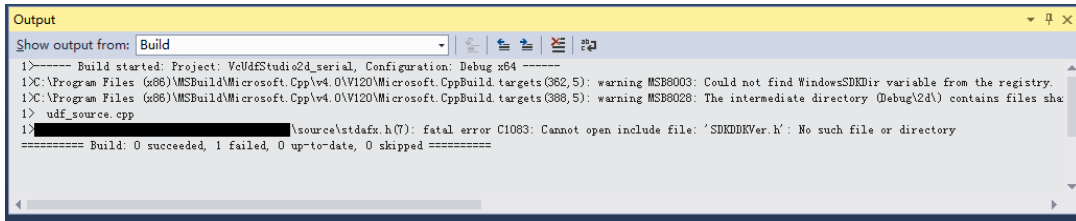


对于 **64 位 Windows**，您只能使用 **64 位 Fluent** 并保证勾选安装“**X64 编译器和工具**”（**X64 compilers and Tools**）。



2. 如果使用 **Visual Studio 2013**，请安装最新的 **Visual Studio 2013 update5**。对于其它更早期的 **2013** 版本，可能出现无法找到“**afxv_cpu.h**”文件，甚至插件菜单混乱的

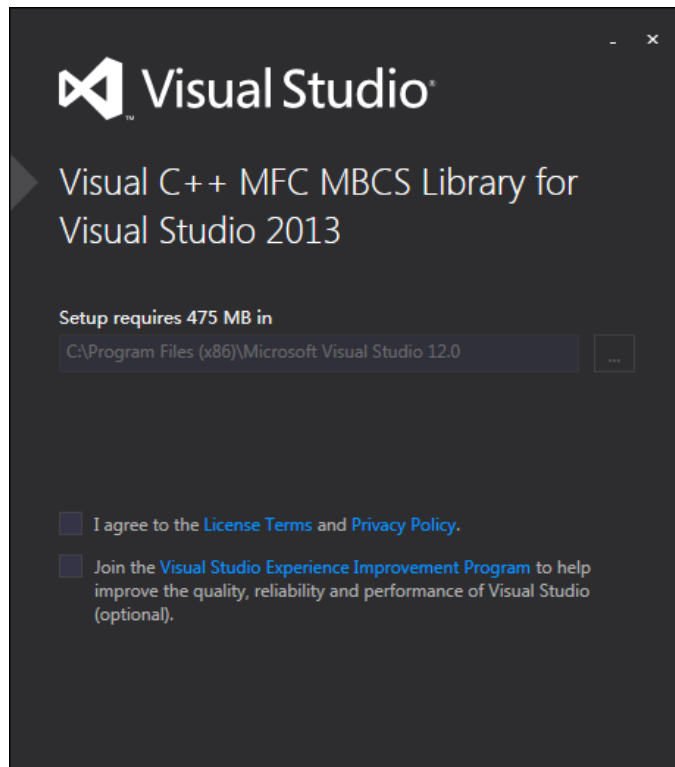
情况。其次，安装前保证网络畅通。否则，可能会报告“WindowSDKDir”变量找不到的错误。



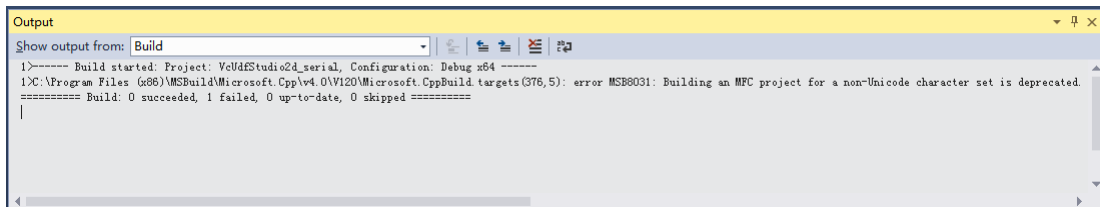
```
Output
Show output from: Build
1>----- Build started: Project: VcUdfStudio2d_serial, Configuration: Debug x64 -----
1>C:\Program Files (x86)\MSBuild\Microsoft.Cpp\v4.0\V120\Microsoft.CppBuild.targets(362,5): warning MSB8003: Could not find WindowsSDKDir variable from the registry.
1>C:\Program Files (x86)\MSBuild\Microsoft.Cpp\v4.0\V120\Microsoft.CppBuild.targets(368,5): warning MSB8028: The intermediate directory (Debug\2d\...) contains files sha
1> udf_source.cpp
1> [redacted]\source\stdafx.h(7): fatal error C1083: Cannot open include file: 'SDKDDKVer.h': No such file or directory
===== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped =====
```

最后，Visual Studio 2013 还需下载安装 Visual C++ MFC 多字节字符库 (Visual C++ MFC Multi-Byte-Character-Set Library)。网址是：

<https://www.microsoft.com/en-US/download/details.aspx?id=40770>

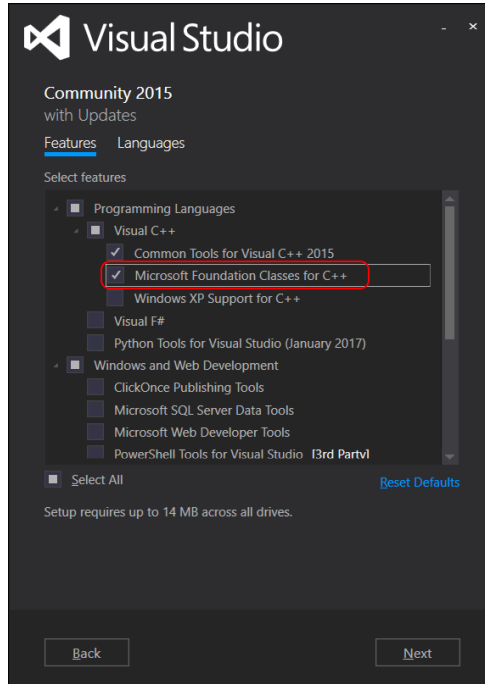


如果 Visual Studio 2013 未安装 Visual C++ MFC 多字节字符库，可能会出现如下错误。

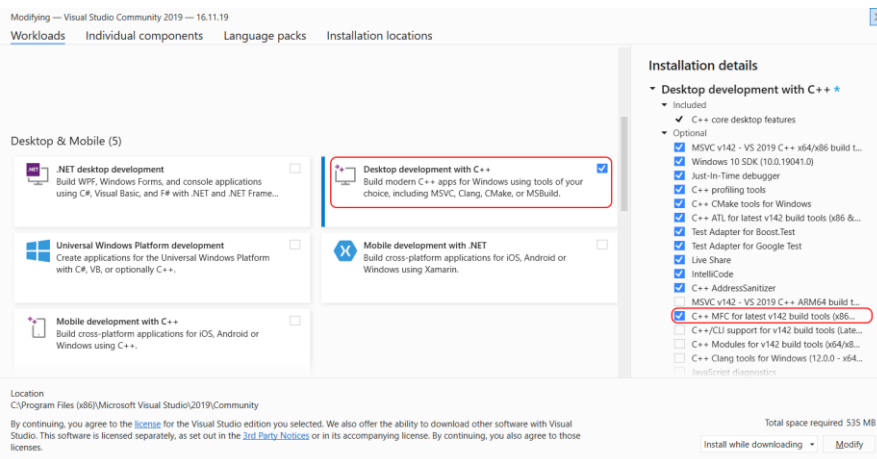
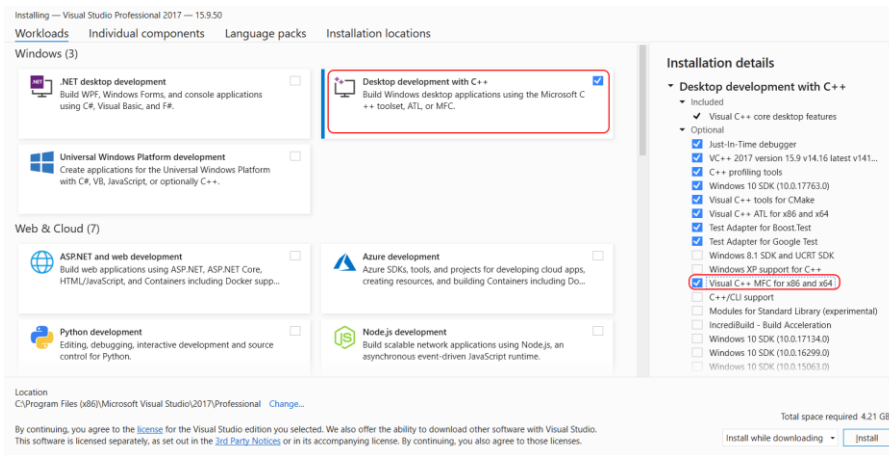


```
Output
Show output from: Build
1>----- Build started: Project: VcUdfStudio2d_serial, Configuration: Debug x64 -----
1>C:\Program Files (x86)\MSBuild\Microsoft.Cpp\v4.0\V120\Microsoft.CppBuild.targets(376,5): error MSB8031: Building an MFC project for a non-Unicode character set is deprecated.
===== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped =====
```

3. 如果使用 Visual Studio 2015 或更高版本，请勾选安装“Microsoft Foundation Classes for C++”组件。

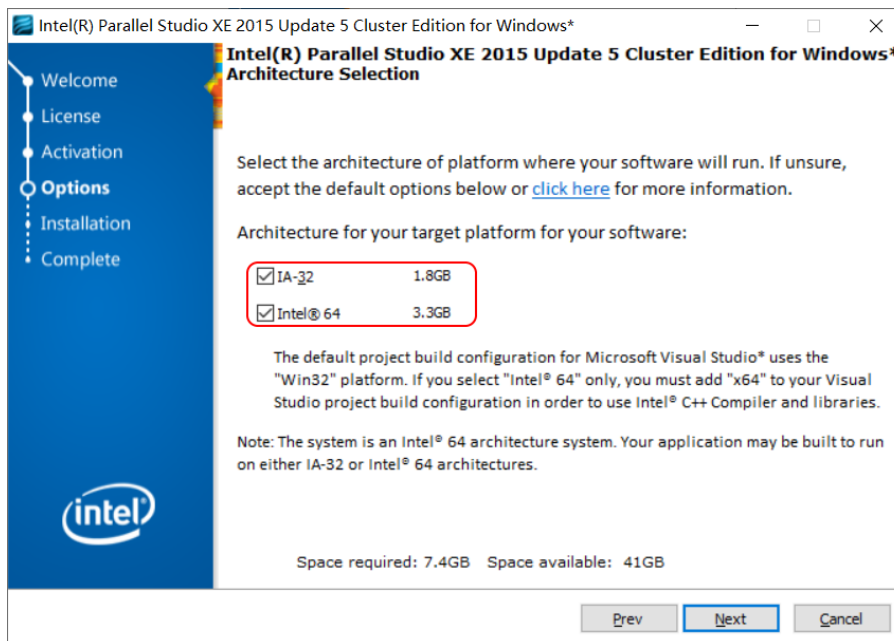
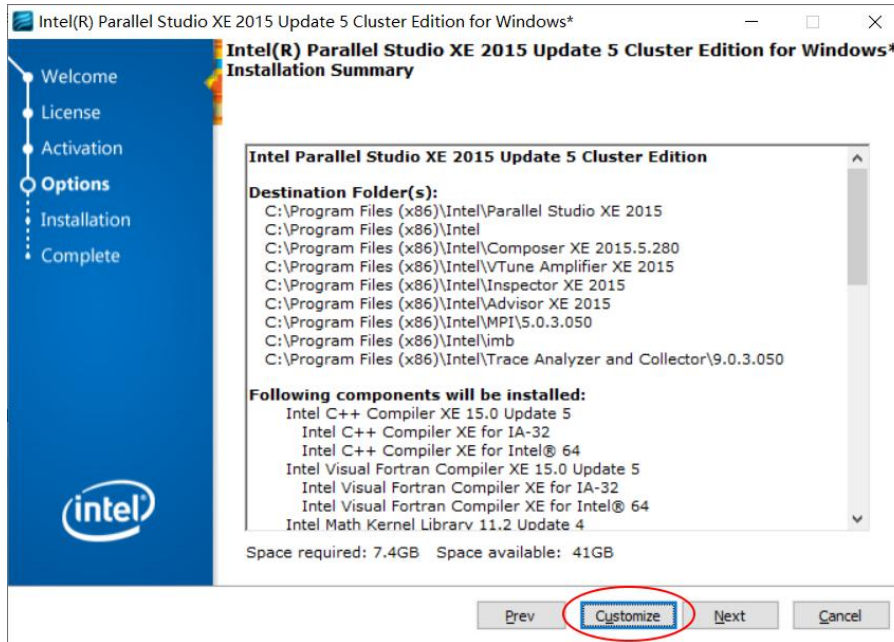


4. 如果使用 **Visual Studio 2017** 或 **2019**，请勾选安装“使用 C++的桌面开发”组件，并勾选“**C++ MFC for x86 & x64**”子模块。

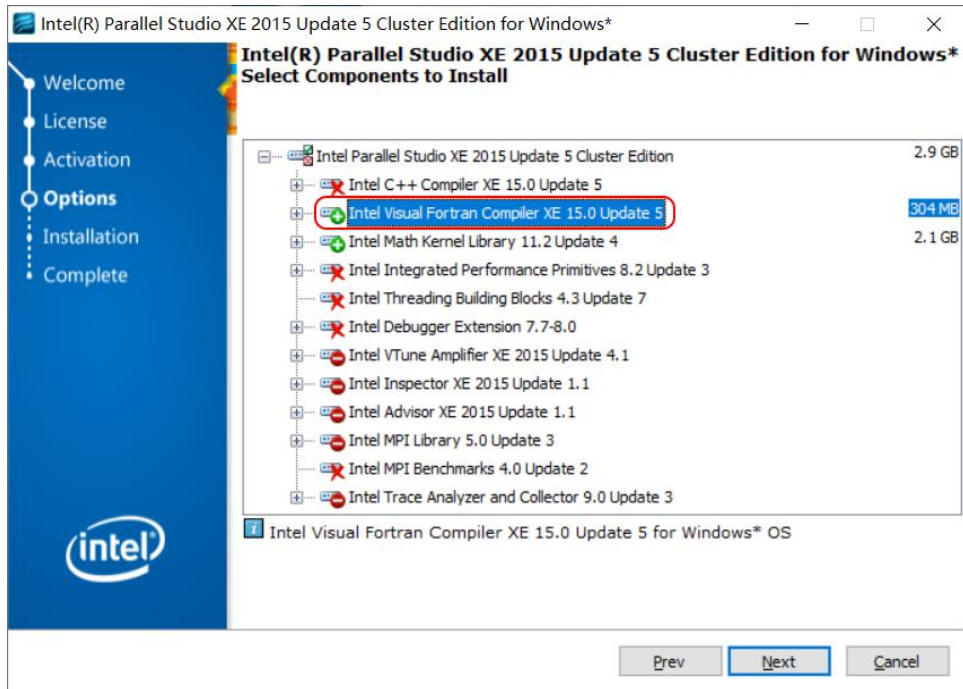


3.2、Intel Visual Fortran 安装注意事项

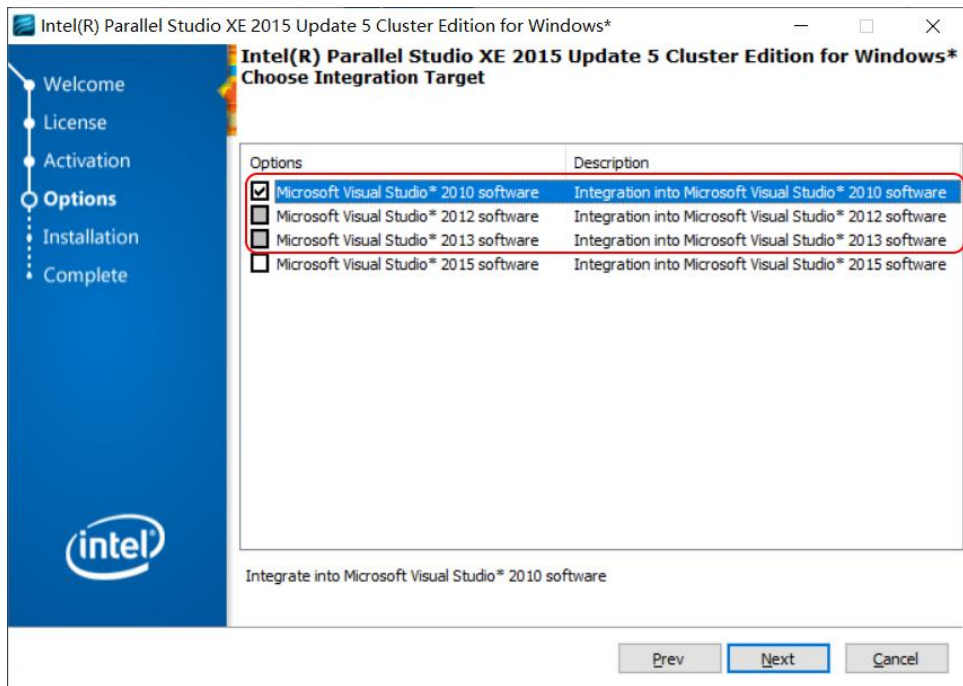
- 如果您想测试调用 **Fortran** 函数功能（或已购买注册版的 **Fortran** 函数功能），则除了 **Visual Studio** 以外还需要另外安装 **Intel Visual Fortran**，支持的版本可参见表 1。注意根据您的 **FLUENT** 的位数（**32** 位还是 **64** 位），确认对应的 **32/64** 位架构编译器已经勾选安装。比如，如果您使用 **64** 位 **Fluent**，那么请安装 **64** 位 **Fortran** 编译器。



- 高版本的 **Intel Visual Fortran** 已经成为软件包 **Intel Parallel Studio XE** 的一个组件，选择自定义安装时请勾选 **Intel Visual Fortran**。其它可根据您的需要选择安装。

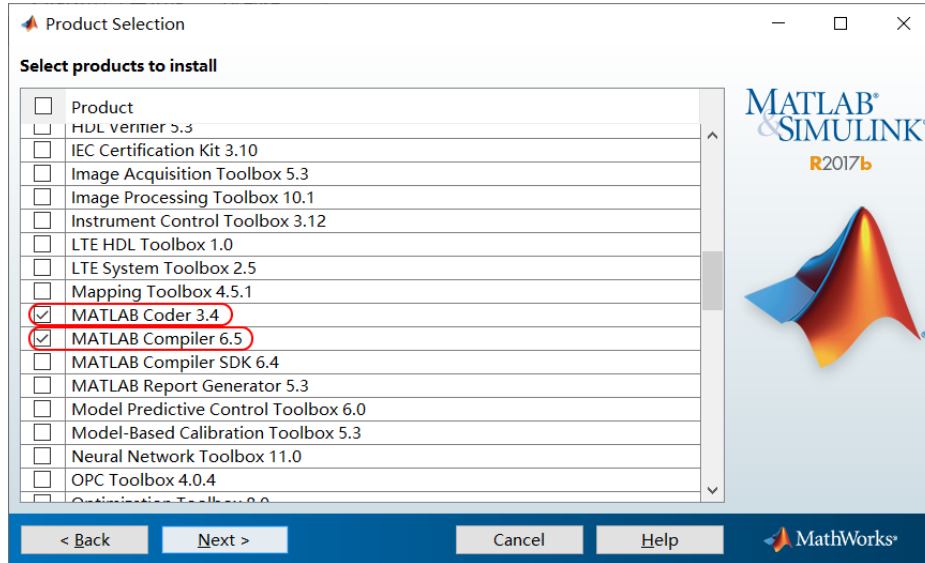


3. 另外，请根据你的 **Visual Studio** 安装版本选择对应的 **Microsoft Visual Studio** 集成选项。 注意，此处可能需要先安装好 **Visual Studio**。



3.3、Matlab 安装注意事项

1. 如果您想测试调用 **Matlab** 函数功能（或已购买注册版的 **Matlab** 函数功能），则除了 **Visual Studio** 以外还需要另外安装 **Matlab**，支持的版本可参见表 1。注意：除了 **Matlab** 自身以外，**MATLAB Coder** 和 **MATLAB Compiler** 是必须勾选安装的，其它根据您的需求安装。

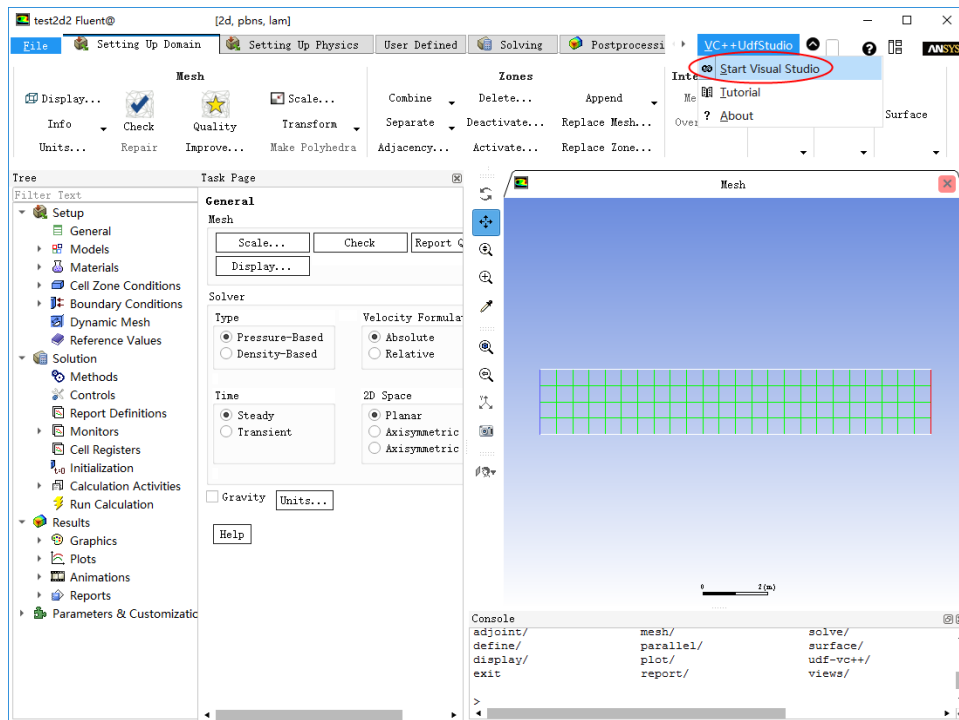


4、UDF 编译调试功能基本使用步骤

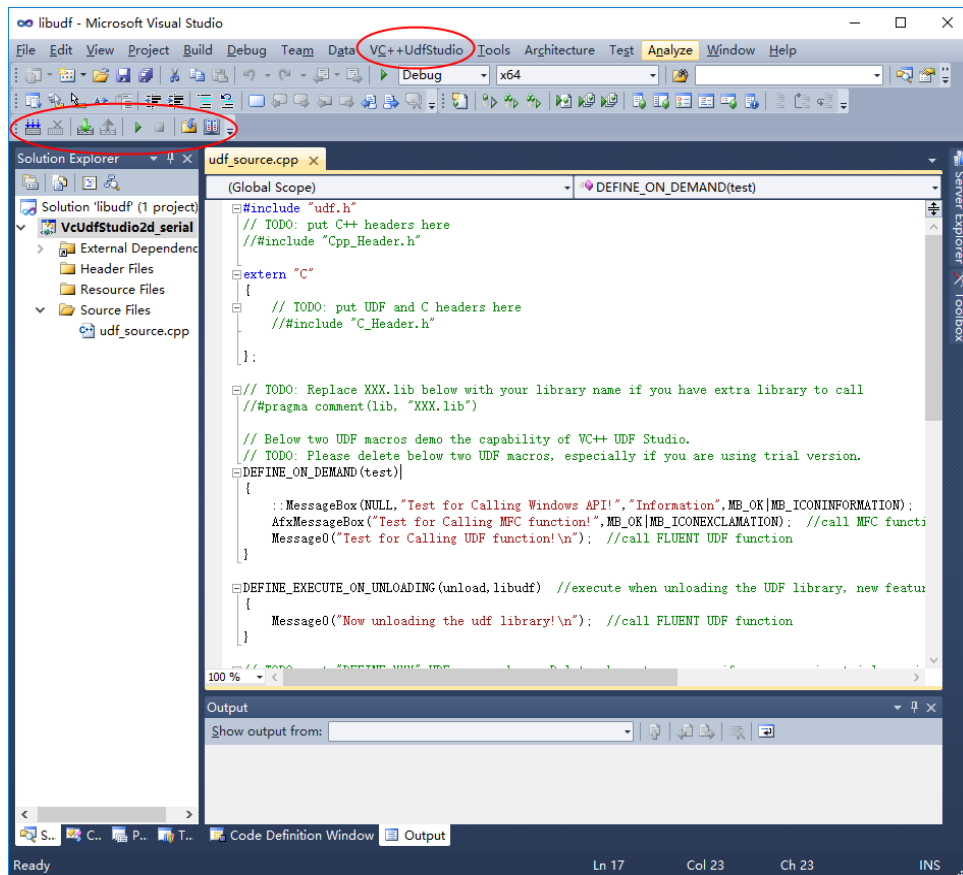
1. 首先安装 **Visual Studio**，确保 **Visual C#**，**Visual C++** 及 “**Visual C++ Tools**” 已勾选（具体注意事项见前一节 “**Visual Studio** 安装注意事项” 内容）。
2. 然后再安装运行 “**VC++ UDF Studio**” 安装包。注意：由于该安装包会搜索已安装的 **Visual Studio** 版本并做相应设置，**请先安装好 Visual Studio 再安装 “VC++ UDF Studio”**。
3. 运行“**VC++ UDF Studio**”加载器并选择欲启动的 **FLUENT** 和 **VC** 版本,然后点击“**OK**”按钮。如果你想运行的 **Fluent** 版本不在列表里，你可以点击 “**Browse**” 按钮来选择 **Fluent** 版本对应的安装目录。

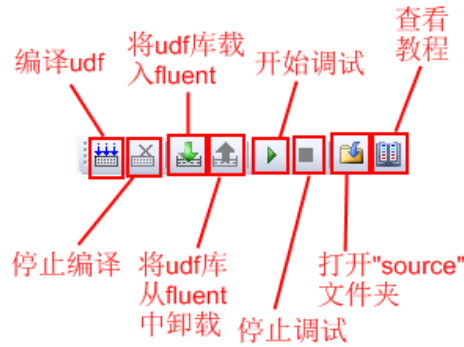


4. 读入一个 **Fluent case** 然后点击 “**Start Visual Studio**” 菜单开始 **UDF** 编程。

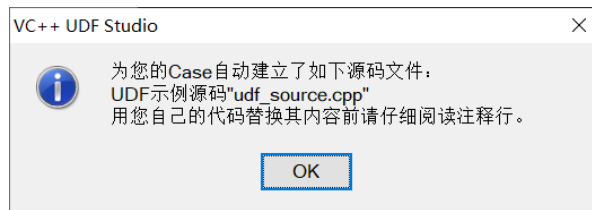


5. 此时，**Visual Studio** 中能看到 “**VC++ UDF Studio**” 的工具栏和菜单。同时，软件会自动在 **case** 的相同目录建立一个名为 “**source**” 的文件夹，包含了所有 **UDF** 源代码。

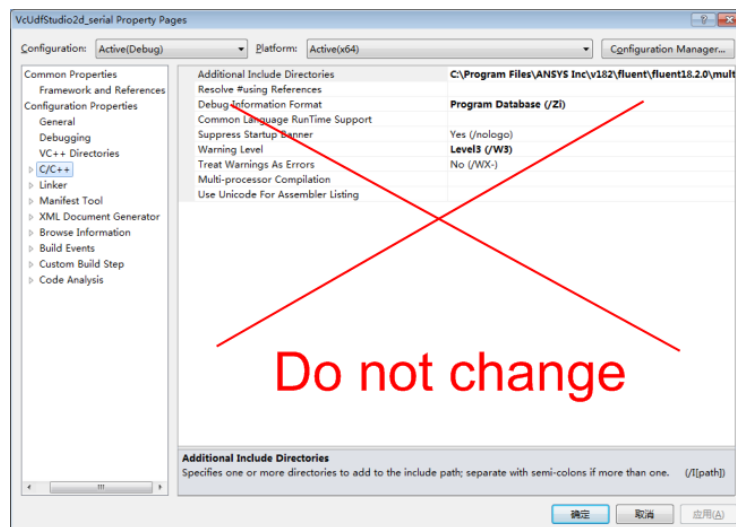




如果“source”目录中未找到 UDF 源代码“udf_source.cpp”文件，则软件会自动为您建立一个示例文件。如果“source”目录中找到该文件，则会直接打开。



Visual Studio 关闭时，项目文件“*.vcxproj”将被自动删除，因此请勿更改项目的设置。如果您想链接额外的库文件“XXX.lib”，您可以在“udf_source.cpp”中增加一句 `#pragma comment(lib, "XXX.lib")`

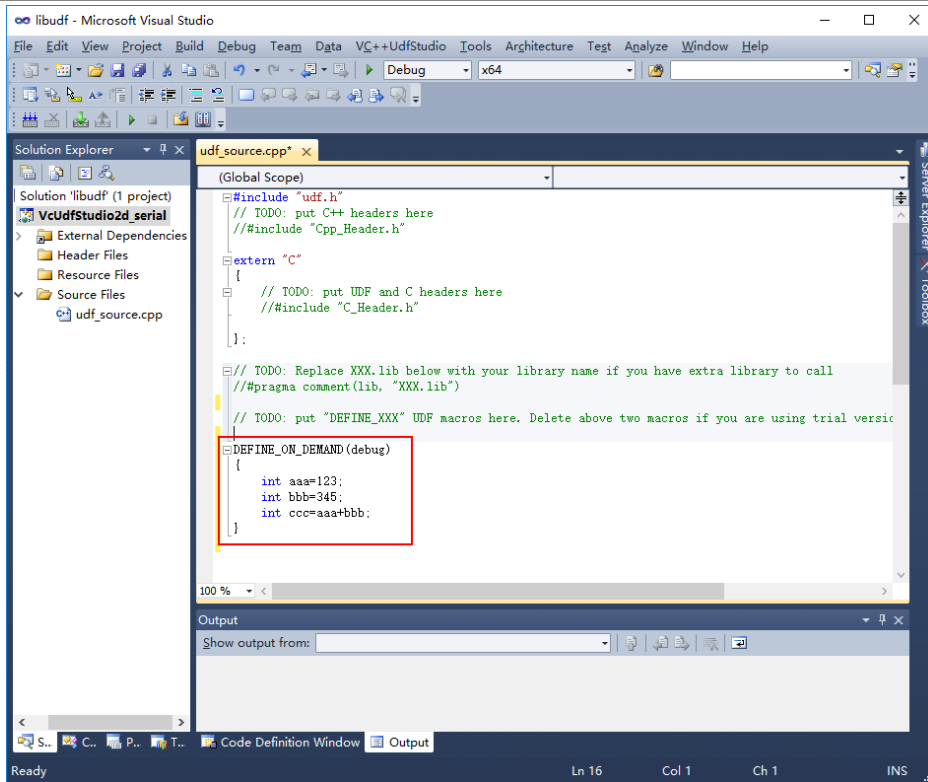


注意: Visual Studio 关闭时，除了必要的源代码，如 udf_source.cpp, for_source.f90, *.m 等文件以外，所有项目文件和临时文件夹（如*.sln, *.suo, *.vcproj, *.vcxproj, *.user, *.filters, *.ncb, *.sdf, Debug 文件夹, Release 文件夹之类）将被自动删除。

6. 编辑 UDF 源码，在“udf_source.cpp”的末尾添加如下测试代码 (如果是试用版，请删除自带的 DEFINE_ON_DEMAND 宏，否则会报告宏总数超过允许数的错误)。

```

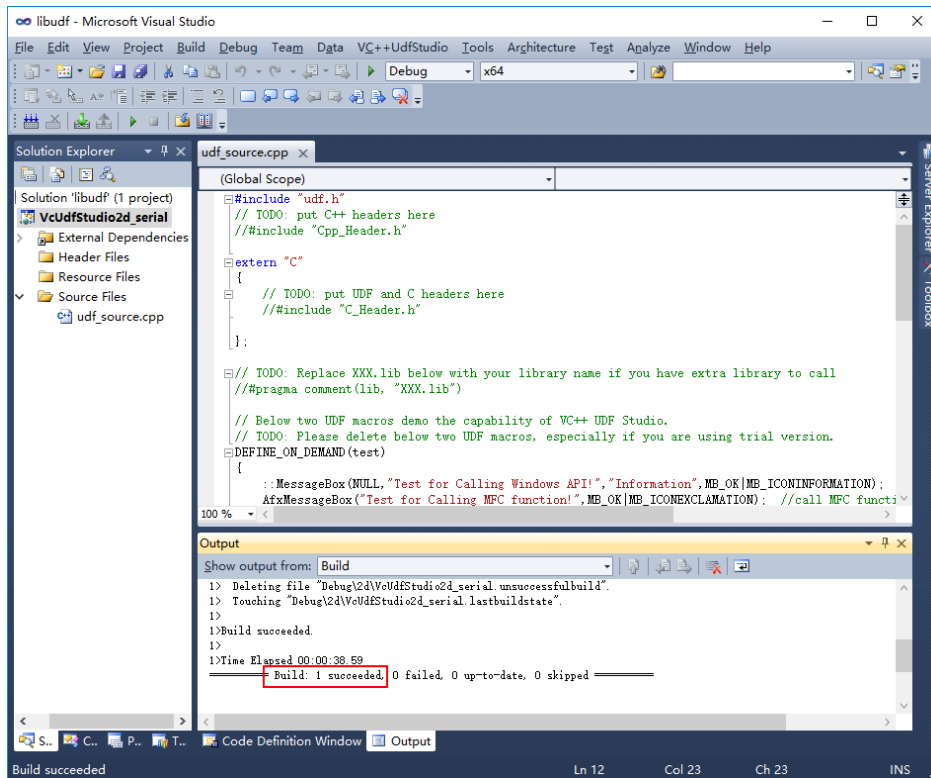
DEFINE_ON_DEMAND(debug)
{
    int aaa=123;
    int bbb=345;
    int ccc=aaa+bbb;
}
    
```



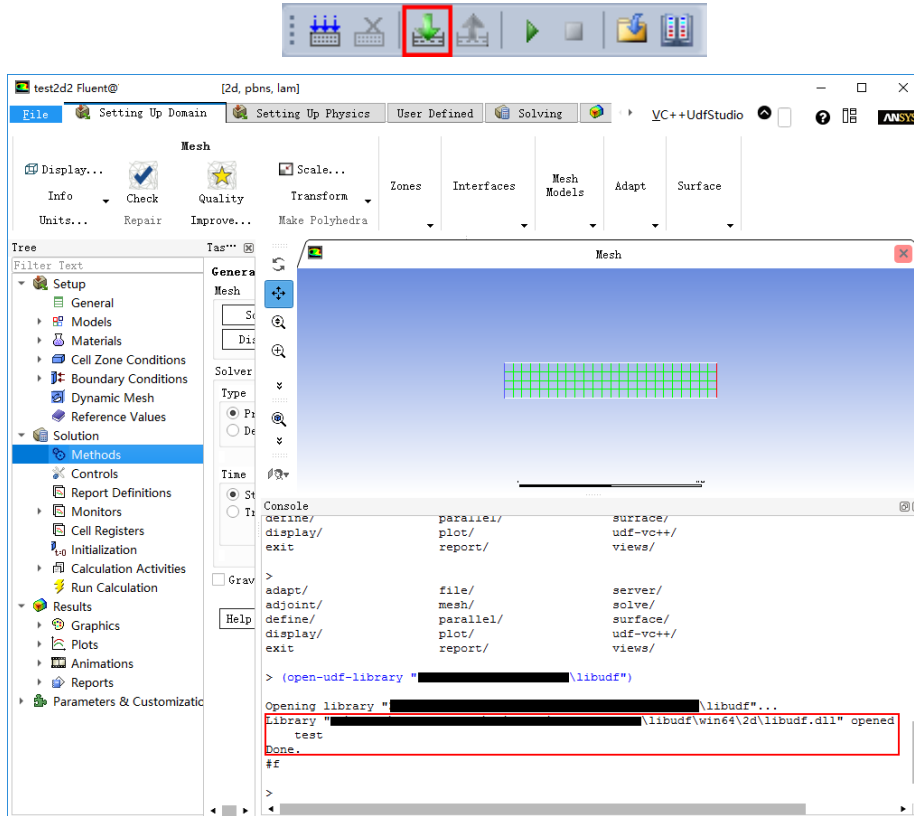
7. 点击“Build UDF library”按钮（或热键“F7”）开始编译UDF源码。



如果没有发现语法错误，则会报告编译成功，如下图所示。只有编译通过后，你才能进行后续的加载或调试。



8. 点击“Load UDF library to Fluent”按钮将编译好的库载入 Fluent。此时，Fluent 控制台界面应该有显示 libudf 库成功加载的响应。

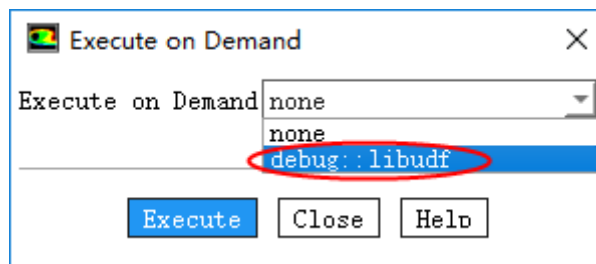


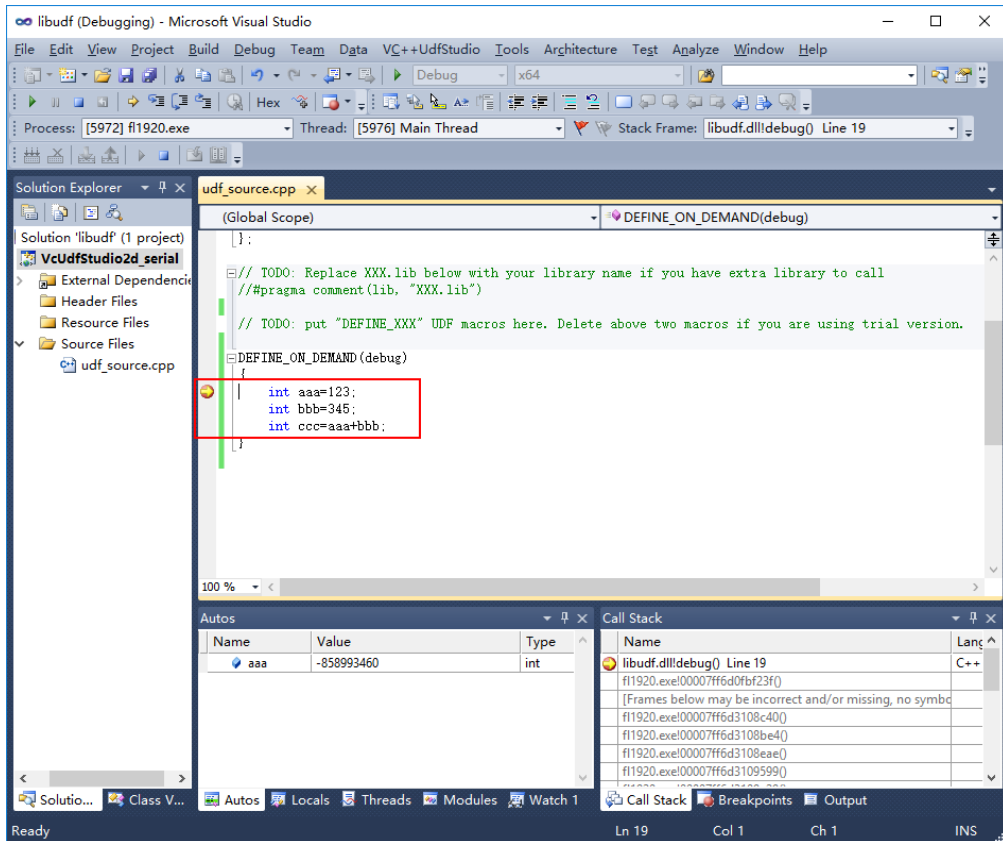
9. 在“int aaa=123;”语句前设置断点（鼠标停在该行热键“F9”），然后点击“Start debugging UDF library”按钮。当然如果不加载直接点该按钮也是可以的，软件会自动先帮您加载，但必须编译通过。



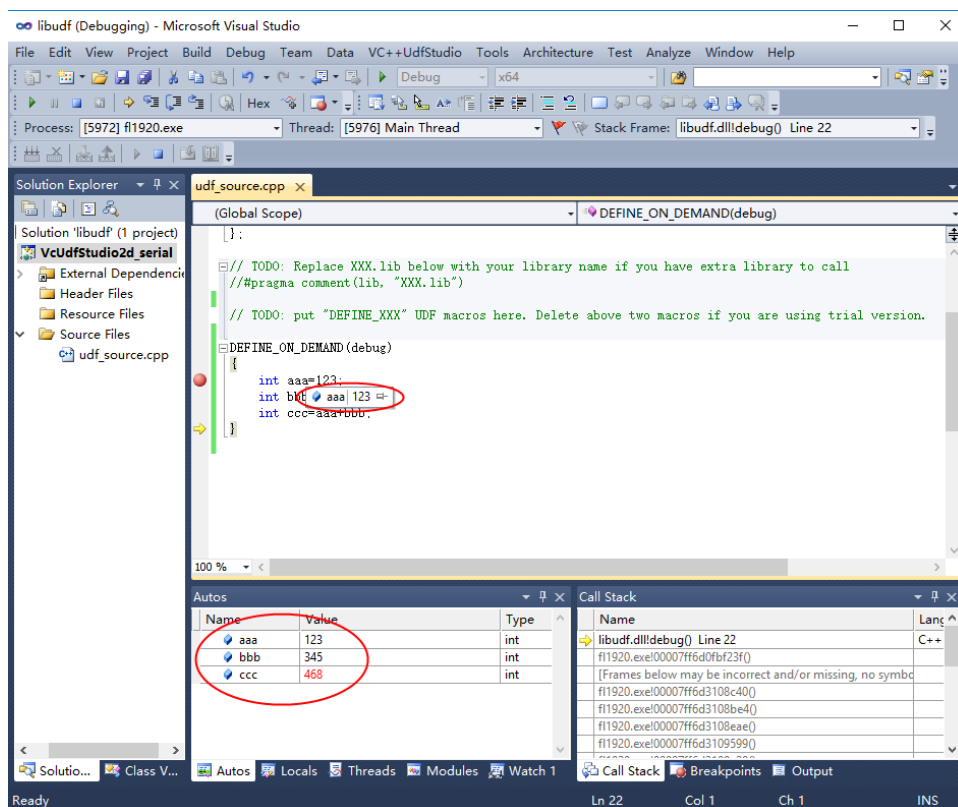
10. 在 Fluent 中执行“debug::libudf”函数，Visual Studio 会自动停在刚才设置的断点处，您可以看到所有变量的值了。注意：有些宏需要下好断点点击调试按钮后再开始 Fluent 迭代计算才能中断到宏内的断点，例如 DEFINE_SOURCE, DEFINE_PROFILE 等等。因为 Fluent 计算过程中才会调用这些宏，断点才会被执行到。

注意：请首先理解你要调试的宏何时被 Fluent 调用。即使在某个宏中下了断点，但 Fluent 未调用该宏，断点也是不起作用的。例如 DEFINE_SOURCE 一般在 FLUENT 迭代计算过程中被调用，DEFINE_INIT 是在初始化时被调用。如果在 DEFINE_SOURCE 宏中下了断点，但没有开始迭代，程序是不会停在断点处的。

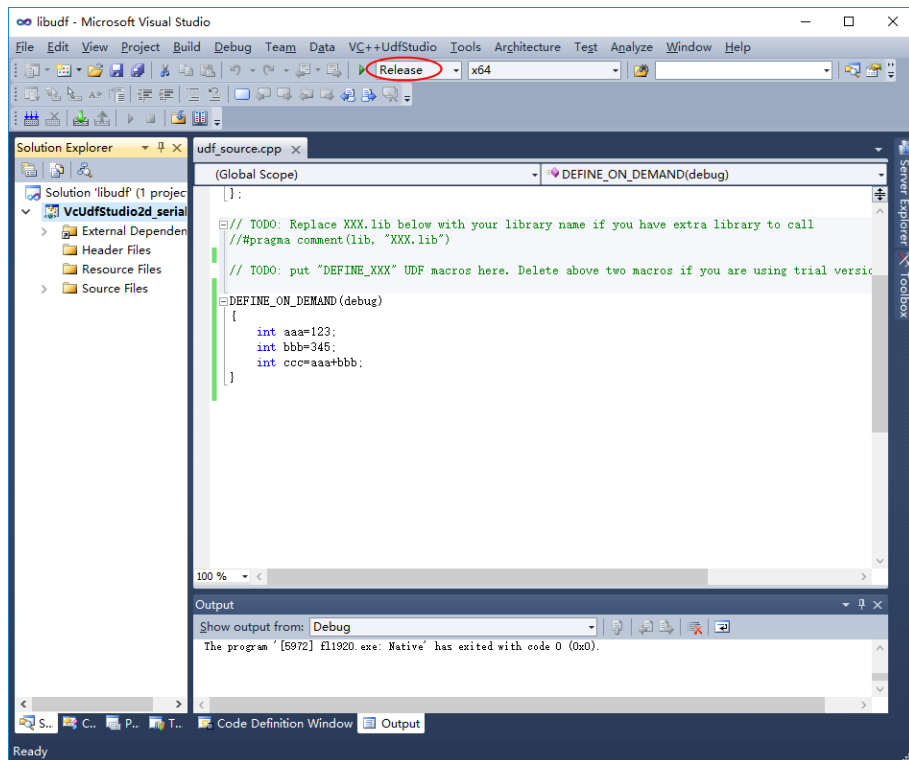




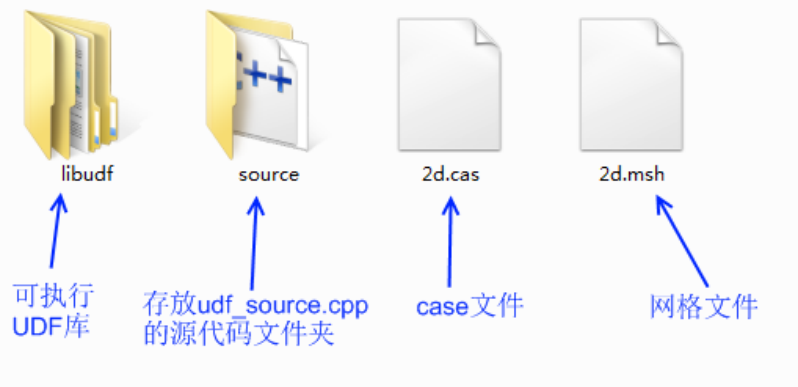
11. 单步调试跟踪（或“F10”热键），您可以采用平时调试 C++ 程序的方式那样观察所有变量的值。



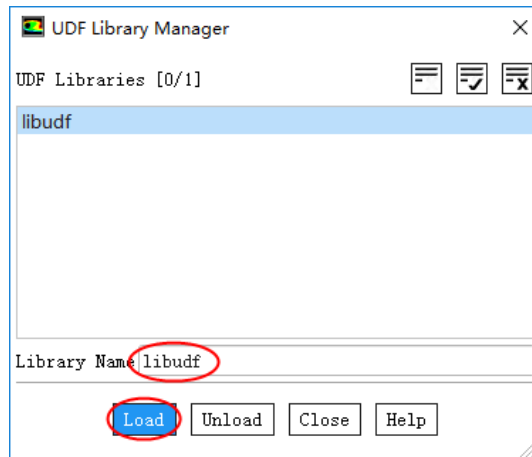
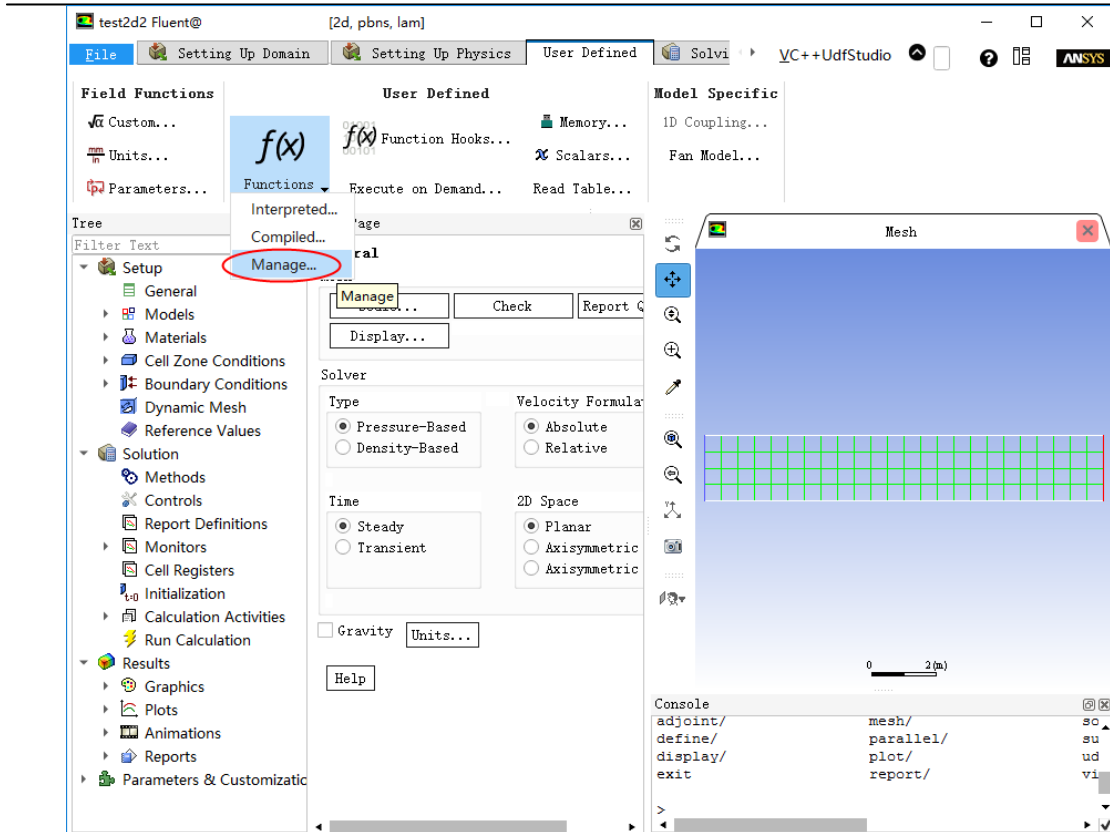
12. 所有 **bug** 修正后，您可以将 **Debug** 模式改为 **Release** 模式，然后在 **Release** 下重新编译一次以便发布您的 **UDF** 库。



13. 现在，您的 **case** 目录一般如下图所示。其中，“**libudf**”文件夹包含了您要发布的 **UDF** 库，而“**source**”文件夹保存了您的 **UDF** 源代码文件“**udf_source.cpp**”。



14. 重新编译通过 **Release** 版本的 **UDF** 库后，您如果想只计算 **case**（不改动或编译 **UDF** 源码）那么可以不必从 **VC++ UDF Studio** 加载器启动 **Fluent**。只要按照通常方式启动 **Fluent**，加载 **UDF** 库可以用“**Define->User-Defined->Functions->Manage**”菜单。在“**Library Name**”编辑框中输入库名“**libudf**”，然后点击“**Load**”按钮即可。**注意**：插件编译时和脱离本插件 **Load** 时候的 **Fluent** 版本、精度版本，**2d/3d** 版本必须一致。此外，脱离插件时高版本 **Fluent** 的默认架构已经是并行架构，试用版插件编译的库因为是真串行，和 **Fluent** 的默认架构不一致，所以会无法 **Load**。你必须用正式版的并行版编译，并保持 **Fluent** 版本、精度版本，**2d/3d** 版本一致，才能直接脱离插件 **Load**。

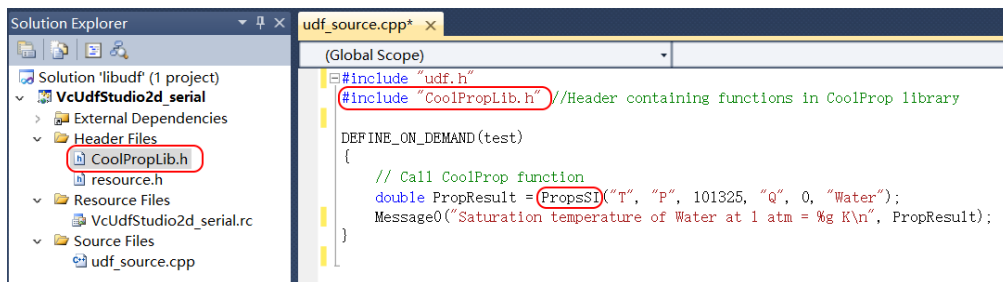


5、调用 CoolProp 函数基本使用步骤

1. 启动器界面中勾选“调用 **CoolProp**”（注意：如果您已经购买注册版本，但未采购“调用 **CoolProp**”这一可选功能，则该选项为禁用状态，要试用该功能可以卸载后重新安装本插件后，自动切换成试用版，从而开放该试用功能）。



2. 启动 **Fluent**，读入 **case**，然后从菜单启动 **Visual Studio**（参见前文“UDF 调试功能基本使用步骤”一节）。此时，程序自动把 **CoolPropLib.h** 加入工程文件夹中，而您则需要将 **#include “CoolPropLib.h”** 语句加入 **udf_source.cpp** 文件中，然后您就可以在源代码 **udf_source.cpp** 中添加 **CoolProp** 函数了。

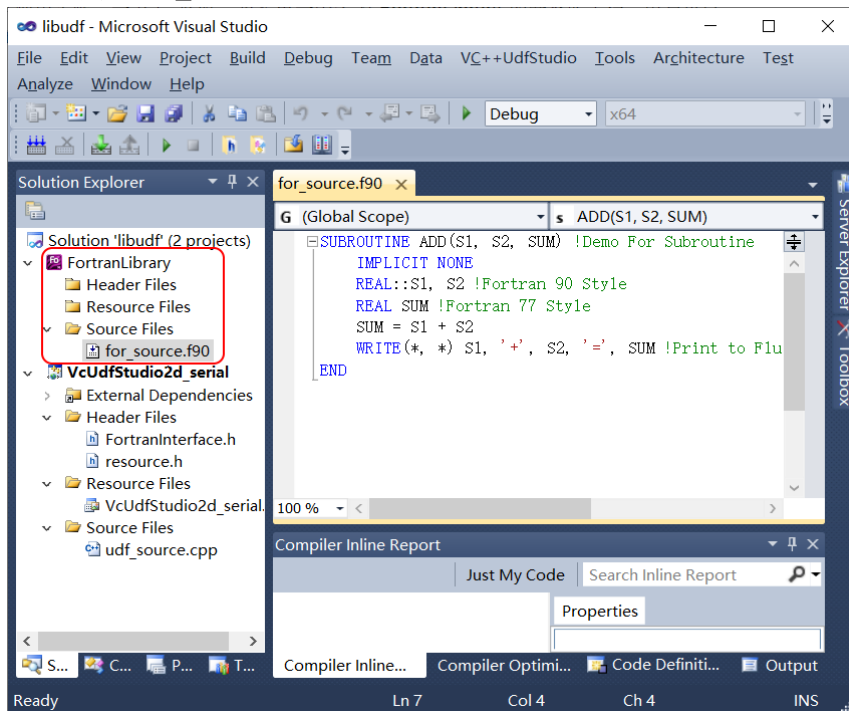


6、调用 Intel Fortran 功能基本使用步骤

1. 安装支持的 **Intel Fortran** 版本（具体注意事项见前一节“Intel Visual Fortran 安装注意事项”内容）。
2. 启动器界面中勾选“调用 **Intel Fortran**”（注意：如果您已经购买注册版本，但未采购“调用 **Intel Fortran**”这一可选功能，则该选项为禁用状态，要试用该功能可以卸载后重新安装本插件后，自动切换成试用版，从而开放该试用功能）。



- 启动 **Fluent**，读入 **case**，然后从菜单启动 **Visual Studio**（参见前文“UDF 调试功能基本使用步骤”一节）。此时，将多出一个名为 **FortranLibrary** 的静态库工程，用户可以在其中的源代码 **for_source.f90** 中添加自己的函数。



- 根据 **for_source.f90** 中的需要调用的函数编辑相应的头文件“**FortranInterface.h**”，该文件为 **Fortran** 函数的 **C** 原型声明类型头文件。表 3 为常见的 **Fortran** 与 **C/C++** 类型的对应关系例子。注意 **Fortran** 函数参数默认为传地址，所以 **C** 原型声明中对应为指针类型。例如自带例子中 **Fortran** 函数为

```

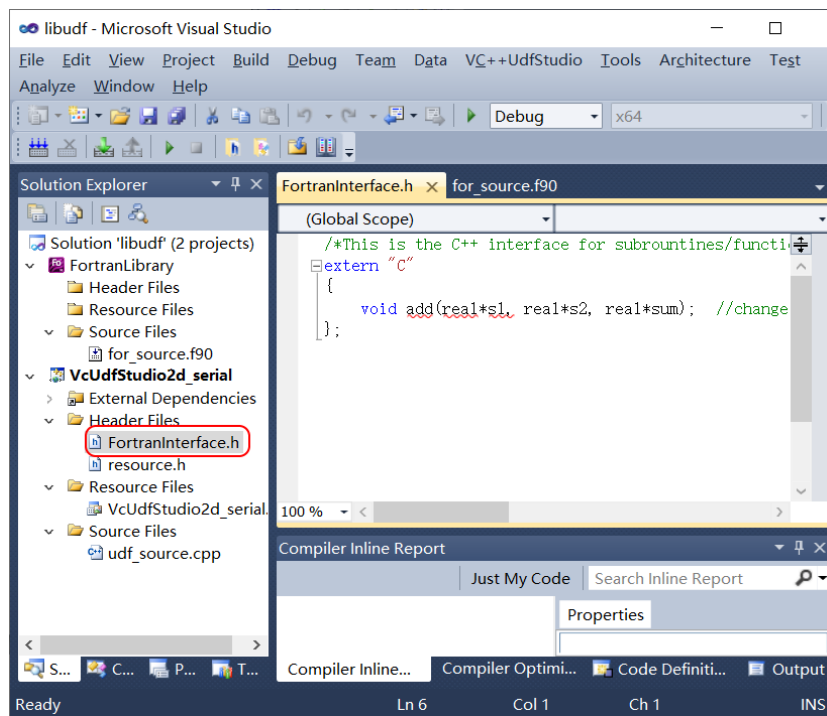
SUBROUTINE ADD(S1, S2, SUM)
  REAL::S1, S2
  REAL::SUM
  
```

对应 **C/C++** 原型声明为

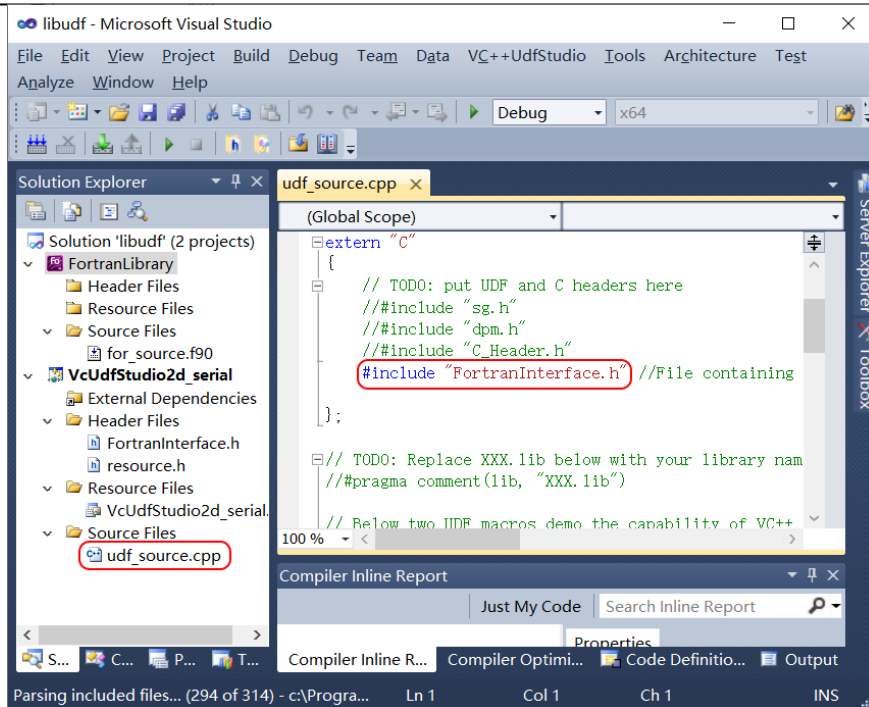
```
void add(real*s1, real*s2, real*sum);
```

表 3、常见 Fortran 和 C/C++ 变量类型的对应关系

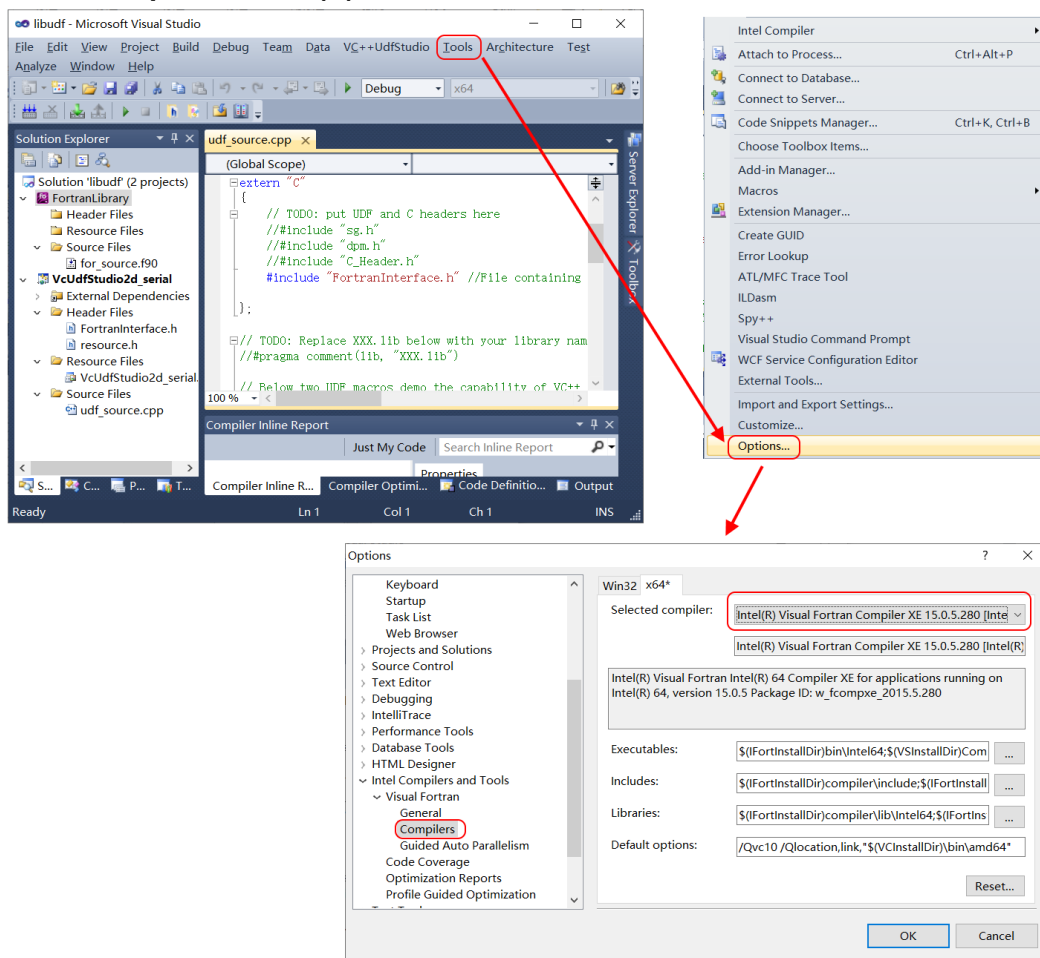
FORTTRAN	C/C++
byte	unsigned char
integer*2	short int
integer	long int or int
integer iabc(2,3)	int iabc[3][2];
logical	long int or int
logical*1	bool (C++, one byte)
real	float (Fluent UDF 中可以是 real 类型)
real*8	double (Fluent UDF 中可以是 real 类型)
real*16	long double
complex	struct{float realnum; float imagnum;}
double complex	struct{double dr; double di;}
character*6 abc	char abc[6];
character*6 abc(4)	char abc[4][6];
parameter	#define <i>PARAMETER value</i>



5. 将#include “FortranInterface.h”语句加入 udf_source.cpp 文件中，这样 UDF 源代码 udf_source.cpp 中就可以调用 Fortran 函数了。

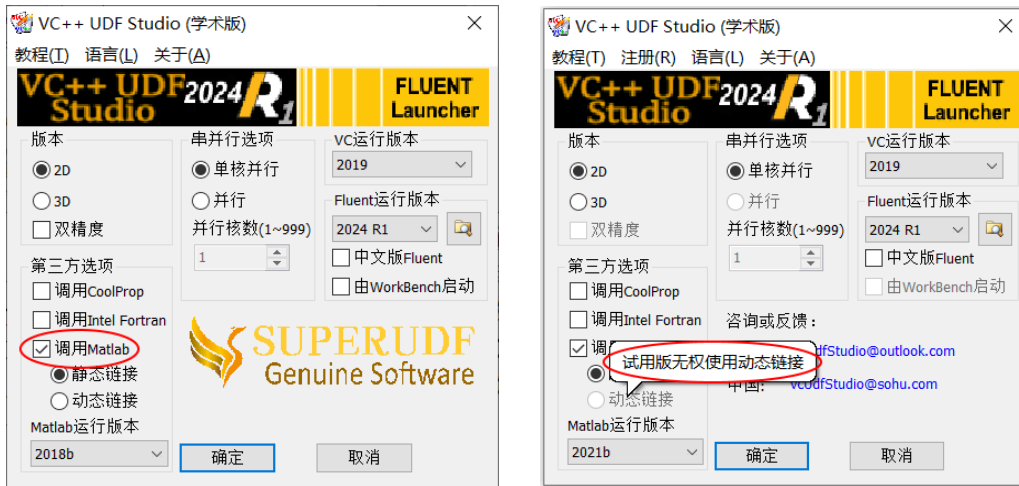


6. 点击编译按钮进行编译。如果你安装了多个版本的 **Fortran** 编译器，则可以在 **Tools->Options->Intel(R) Visual Fortran** 菜单中设置编译时需要采用的版本。



7、调用 Matlab 功能基本使用步骤

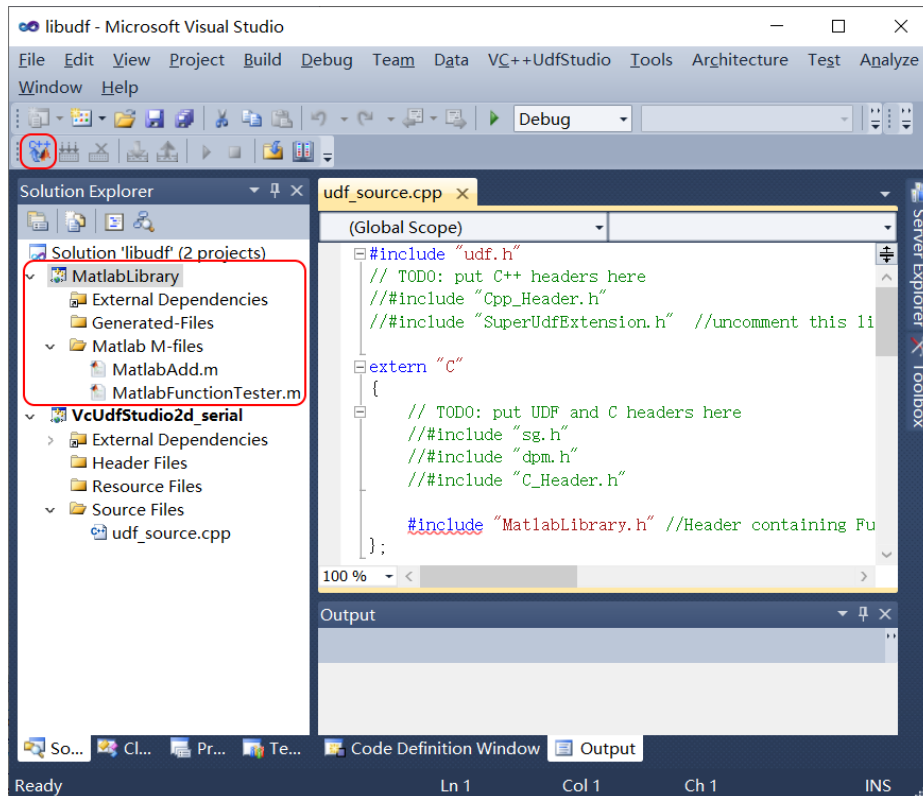
1. 安装支持的 **Matlab** 版本（具体注意事项见前一节“**Matlab** 安装注意事项”内容）。
2. 启动器界面下勾选“调用 **Matlab**”，并选择需要运行的版本（注意：如果您已经购买注册版本，但未采购“调用 **Matlab**”这一可选功能，则该选项为禁用状态，要试用该功能可以卸载后重新安装本插件后，自动切换成试用版，从而开放该试用功能）。此外，如果您想修改已有的 **Matlab** 运行版本，可能需要管理员权限。



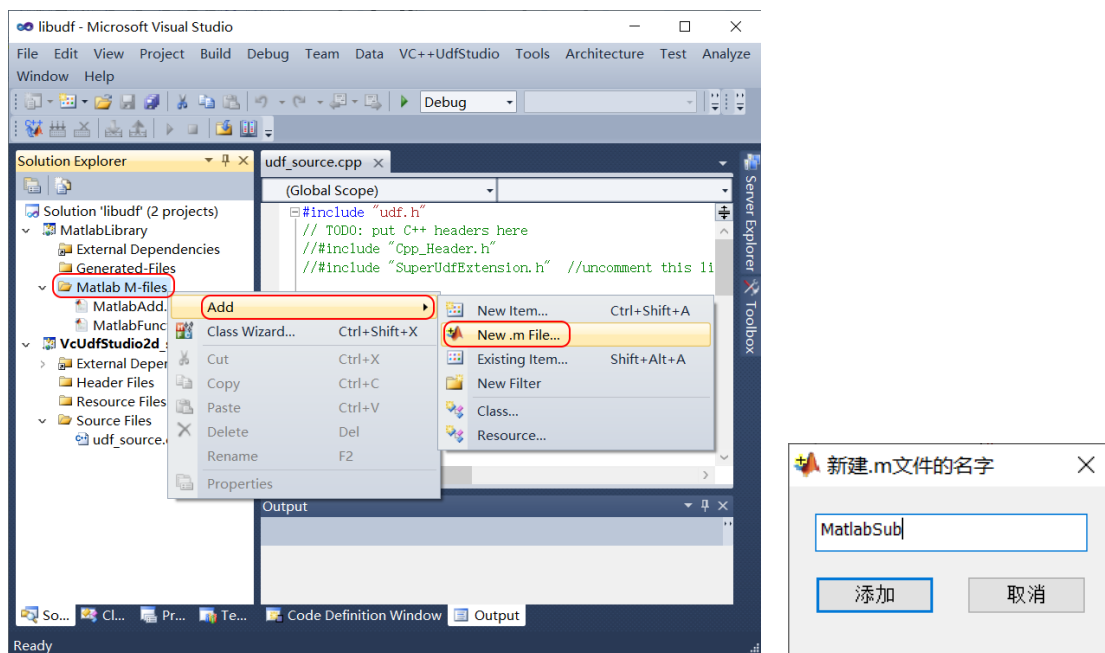
3. 选择静态链接或动态链接子选项，两者的区别在于：静态链接编译生成 **UDF** 库以后不依赖于 **Matlab** 运行库，可以在未安装相应 **Matlab** 版本的机器上运行，但缺点是有些 **Matlab** 函数是不支持的，例如 **GridData**, **ode45**, **plot**, **eval** 等。此时可以改用动态链接，基本所有 **Matlab** 函数都支持，但是如果在新的机器上运行编译好的 **UDF** 库，需要安装对应版本的 **Matlab** 运行库。值得庆幸的是，使用 **VC++UDFStudio** 软件的本机上因为已经安装有 **Matlab** 软件包，无需额外安装 **Matlab** 运行库。

注意：试用版中动态链接模式已禁用，只有采购解锁后才能使用。另外一个已知问题是，“动态链接”模式下某些版本 **Fluent** 和有些 **Matlab** 版本存在冲突，会导致初始化函数 **MatlabLibraryInitialize** 执行失败。此时，您可以安装 **Fluent 2023R2** 和 **Matlab 2018b**。该组合搭配经测试不存在冲突问题。

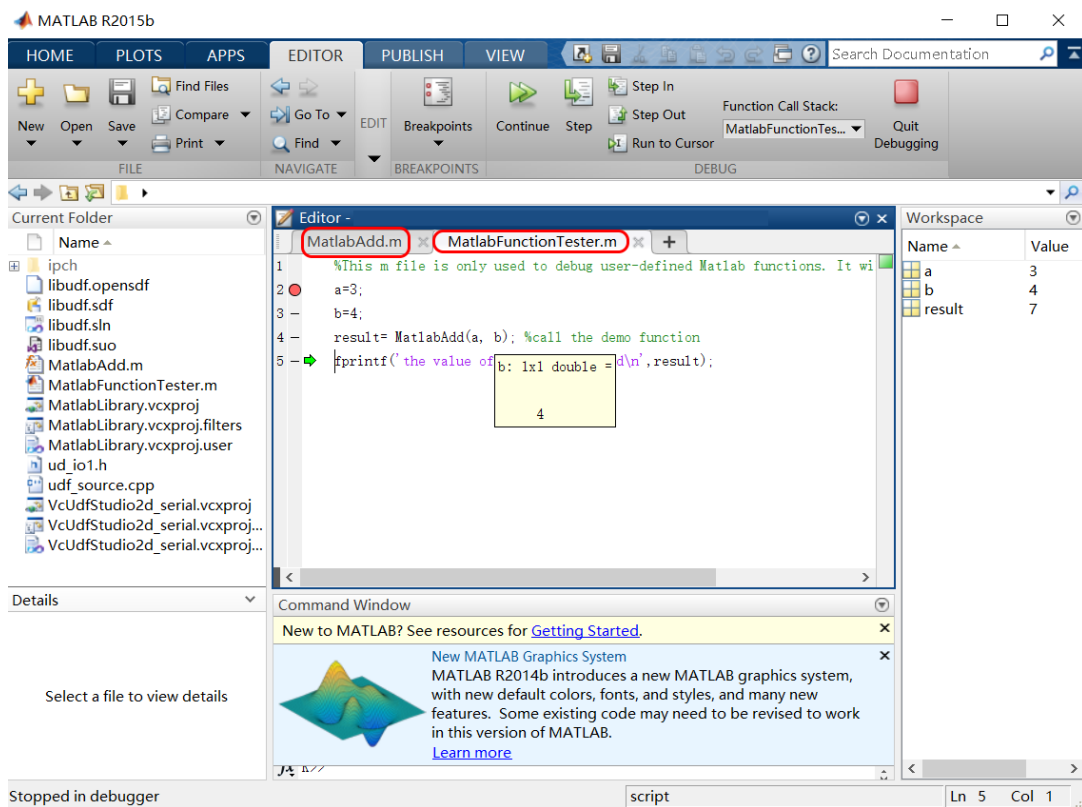
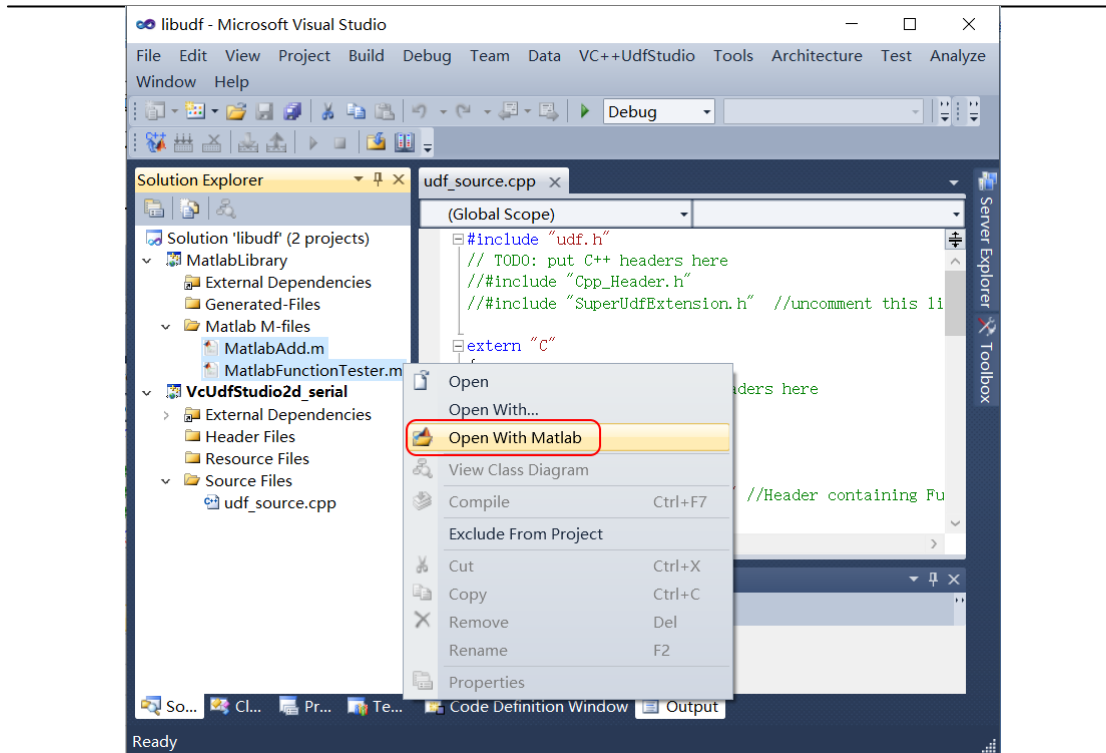
4. 启动 **Fluent**，读入 **case**，然后从菜单启动 **Visual Studio**（参见前文“**UDF** 调试功能基本使用步骤”一节）。此时，将多出一个“将 **.m** 文件转为 **C/C++**”的工具栏按钮和一个名为 **MatlabLibrary** 的静态库工程，用户可以在“**Matlab M-files**”文件夹中添加自己的 **Matlab** 函数。注意，其中 **MatlabFunctionTester.m** 文件是用来在 **Matlab** 中调试测试 **Matlab** 函数用，请勿删除（后面会具体操作说明）。



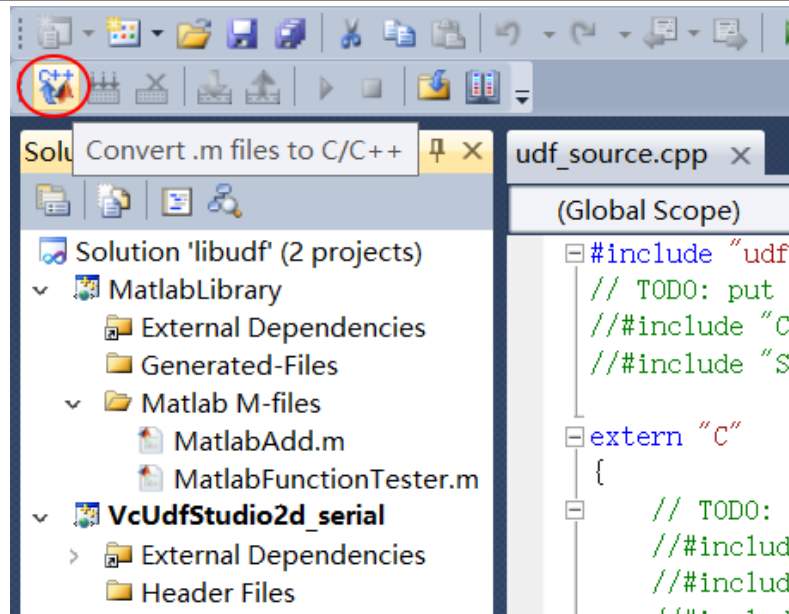
5. 在“**Matlab M-files**”文件夹上右键弹出菜单可以添加新的 **m** 文件，即新的 **Matlab** 函数（由于 **Matlab** 采用一个函数一个 **m** 文件形式）。



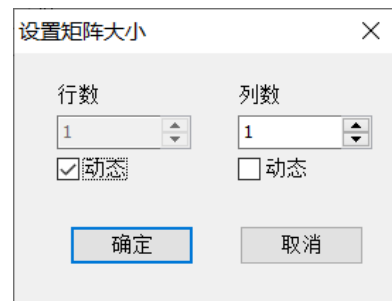
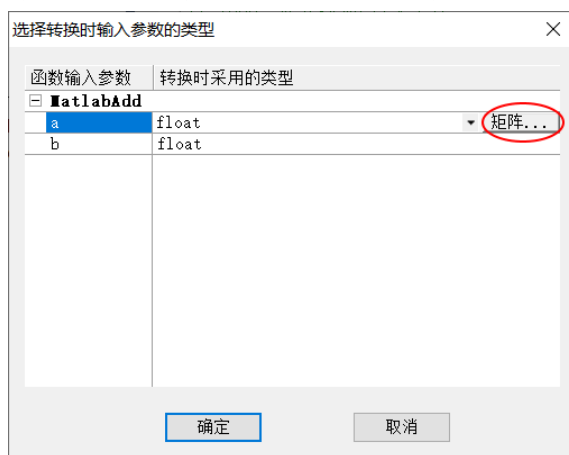
6. 选中需要的 **m** 文件，右键弹出菜单选择“用 **Matlab** 打开”，可以在 **Matlab** 中利用 **MatlabFunctionTester.m** 对用户定义的 **Matlab** 函数进行调用并调试排除错误，保证没有错误以后关闭 **Matlab**。



7. 点击“将 .m 文件转为 C/C++”按钮，准备将所有 Matlab 函数（除了 MatlabFunctionTester.m 以外）转为 C/C++ 文件。注意：“静态链接”模式下，不支持转换的 Matlab 函数在尝试转换结束时会有提示。

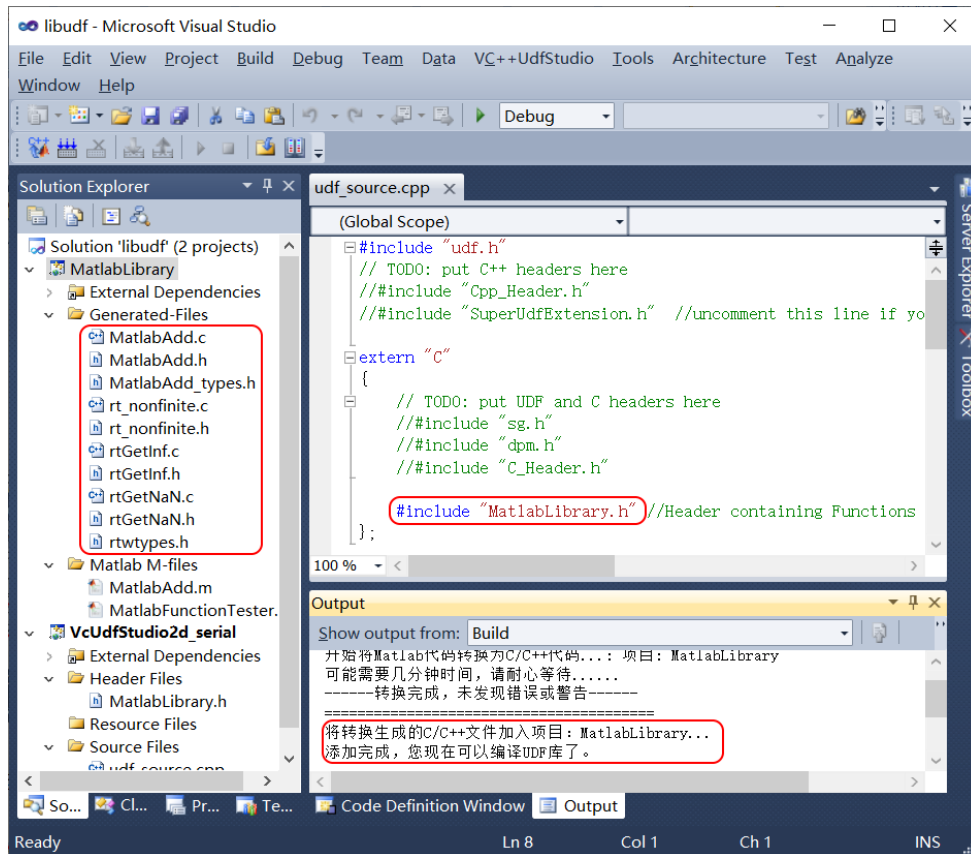


- （仅“静态链接”模式下）在下拉框中选择 **Matlab** 函数的输入参数类型，默认为标量非矩阵，如果需要额外点击“矩阵”按钮，弹出对话框设置矩阵的行列尺寸，若为动态数组请勾选“动态”框。点击“确定”开始转换。

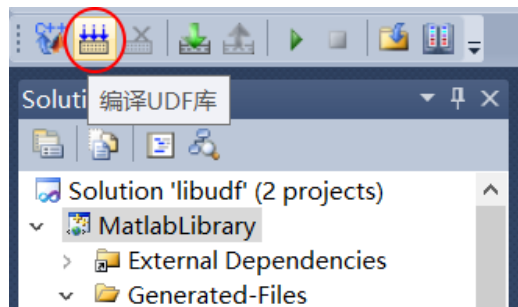


- 如果转换成功，软件自动将产生的 **C/C++** 文件加入到“**Generated-Files**”文件夹。所有可以调用的 **Matlab** 函数原型声明都在 **MatlabLibrary.h** 文件中。

注意：在 **UDF** 源代码“**udf_source.cpp**”中调用 **Matlab** 函数之前，若是“静态链接”模式下，请确认 **extern "C"** 的大括号中已经加入 **#include "MatlabLibrary.h"** 的语句。如果是“动态链接”模式下，**#include "MatlabLibrary.h"** 语句请放置在 **extern "C"** 的大括号外面。

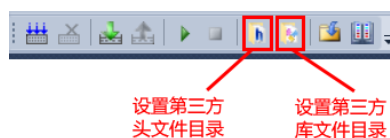


10. 点击“编译”按钮产生 UDF 库，后续其它操作同前文“UDF 编译调试功能基本使用步骤”。

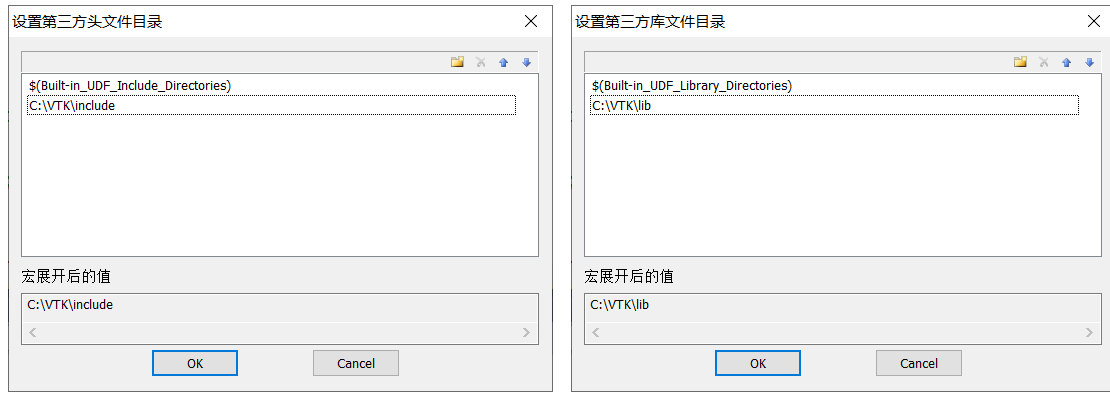


8、设置第三方库目录（仅企业版）

1. 企业版用户可以用如下两个按钮设置第三方头文件目录和库文件目录。



2. 点击按钮后，分别出现如下对话框，可以通过手动或浏览加入第三方头文件和库文件的目录。
\$(Build-in_UDF_Include_Directories) 和 **\$(Build-in_UDF_Library_Directories)** 分别代表 VC++UdfStudio 所自带需要的头文件和库文件的目录，不允许修改，但可以调整它们和额外第三方库的前后顺序。

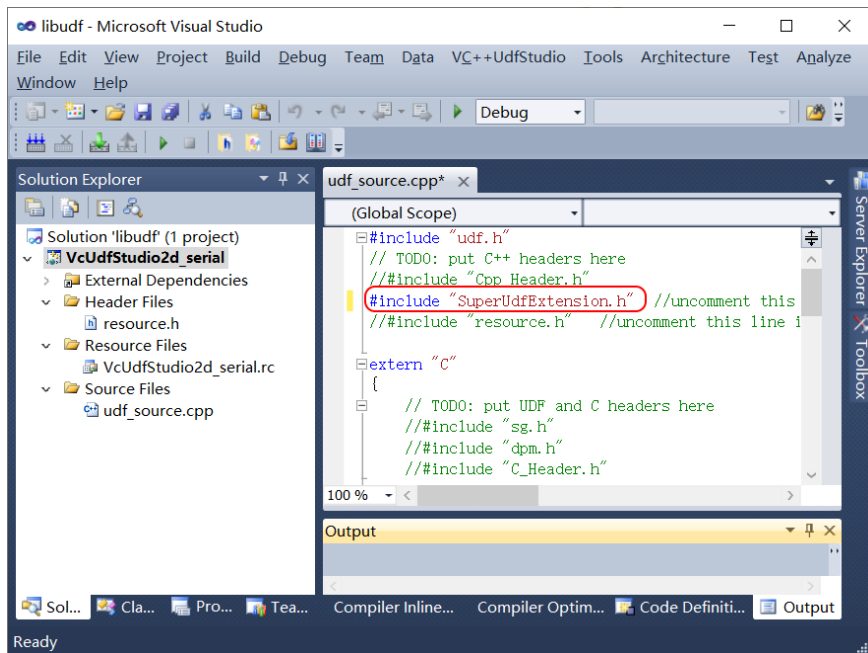


9、拓展函数库

9.1、开启拓展函数库

1. 本工具已将一些常用功能以第三方库的形式封装好供用户直接调用，从而大大拓展 **UDF** 的功能。如下图所示，如需调用拓展的第三方函数，用户只需要将如下语句去掉注释。

#include "SuperUdfExtension.h"



9.2、拓展函数详解

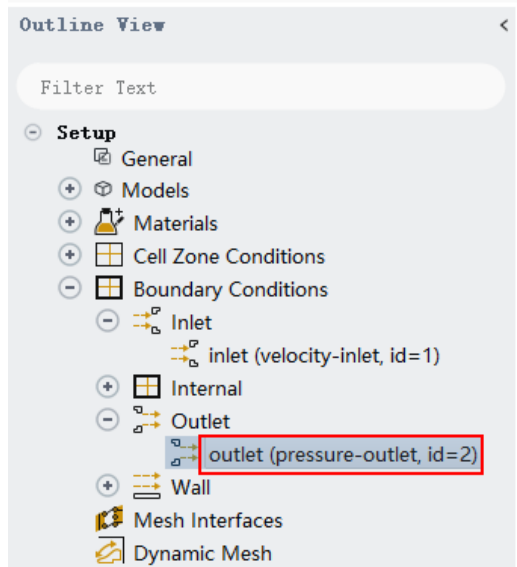
1. void SuperUdf_Initialize(HMODULE hLibudfDllModule)

此函数负责拓展库的初始化，hLibudfDllModule 为 udf 库的模块句柄，可以使用 AfxGetInstanceHandle() 获取（参见下一节实例）。

注意：此函数必须在调用其它拓展函数前调用。较佳的调用该函数的地方是 **DEFINE_EXECUTE_ON_LOADING** 宏中。

2. `int SuperUdf_GetZoneIdByName(char* strZoneName)`

根据 **zone** 或边界的名字来获取对应的 **zone ID**，如下图所示的 **case** 的 **UDF** 源码中调用 `SuperUdf_GetZoneIdByName("outlet")` 函数将返回 2。



该函数主要用来解决源码通用性问题，由于 **UDF** 中经常会调用 `LookUp_Thread(domain, zone_ID)` 来获取 **Thread**，里面的 **zone_ID** 是关键参数，但是会随着网格变化而变化，很多用户每次更换网格后只能手动查到对应 **zone ID** 后修改源码再重新编译来实现，十分不方便。有了此函数后只需要画网格时取固定名字，该函数会自动根据名字获取其 **ID**。

注意：此函数只能在 **serial** 或 **host** 上调用，**node** 上调用会返回-1。但可以在 **serial** 或 **host** 上获取以后再调用 `host_to_node_int` 系列函数将值传给 **node**。

3. `void SuperUdf_Interrupt();`

该函数用来中断正在进行的稳态或非稳态迭代。你可以将该语句放在 **DEFINE_ADJUST** 或 **DEFINE_EXECUTE_AT_END** 中，这样在 **UDF** 代码中一旦达到你要的标准你就可以马上调用该函数停止迭代计算。

4. `HWND SuperUdf_GetFluentMainWnd();`

获得 **Fluent** 的主窗体句柄（仅企业版，详见[编程手册](#)）

5. `void SuperUdf_Steady_Iterate(int nTimes)`

驱动 **Fluent** 进行稳态迭代 **n** 步（仅企业版，详见[编程手册](#)）

6. `void SuperUdf_ExecuteConsoleCommand(char* strAnsiConsoleCommand)`

驱动 **Fluent** 执行 **TUI** 或 **Scheme** 命令（仅企业版，详见[编程手册](#)）

7. `void SuperUdf_AddUserMenu(UINT uMenuResourceID)`

在 **Fluent** 中插入用户菜单（仅企业版，详见[企业版编程手册](#)）

8. `void SuperUdf_EnableMenuItem(UINT uTargetMenuID, BOOL bEnabled)`

将用户菜单变灰禁止或恢复可用（仅企业版，详见[编程手册](#)）

9. void SuperUdf_SetMenuBmpAndFun(MenuItemBmpAndFun menuBmpAndFuns[], ULONG nCount)

设置用户菜单对应的位图和点击响应函数（仅企业版，详见[编程手册](#)）

10. void SuperUdf_SetMenuSelectCallBack(MENUSELECTPROC UserCallBackFunction)

设置选择菜单时响应回调函数，可在此回调函数中动态使菜单变灰或恢复（仅企业版）

9.3、学术版拓展函数实例

以下为学术版中拓展函数的编程实例（企业版拓展函数实例见[编程手册](#)）。

```
#include "udf.h"
#include "SuperUdfExtension.h"

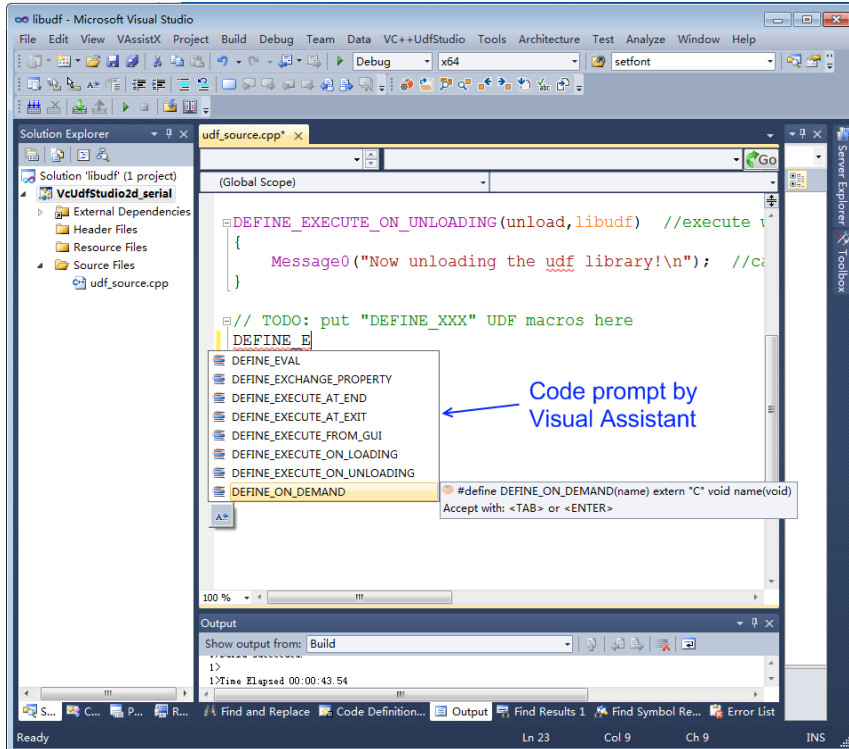
DEFINE_ON_DEMAND(GetOutletId)
{
    int outlet_id;
    face_t f;
    Thread*tf;
    Domain*domain=Get_Domain(1);
#ifdef !RP_NODE
    outlet_id=SuperUdf_GetZoneIdByName("outlet"); //get the id of zone whose name is "outlet"
#endif
    host_to_node_int_1(outlet_id);

#ifdef !RP_HOST
    if(-1==outlet_id)
        Message("Can't get the ID on myid=%d\n",myid);
    else
    {
        tf=Lookup_Thread(domain, outlet_id);
        Message("myid=%d, outlet id=%d\n",myid, outlet_id);
        begin_f_loop(f,tf)
        {
            if(PRINCIPAL_FACE_P(f,tf))
            {
                // loop over faces on "outlet"
            }
        }
        end_f_loop(f,tf)
    }
#endif
}

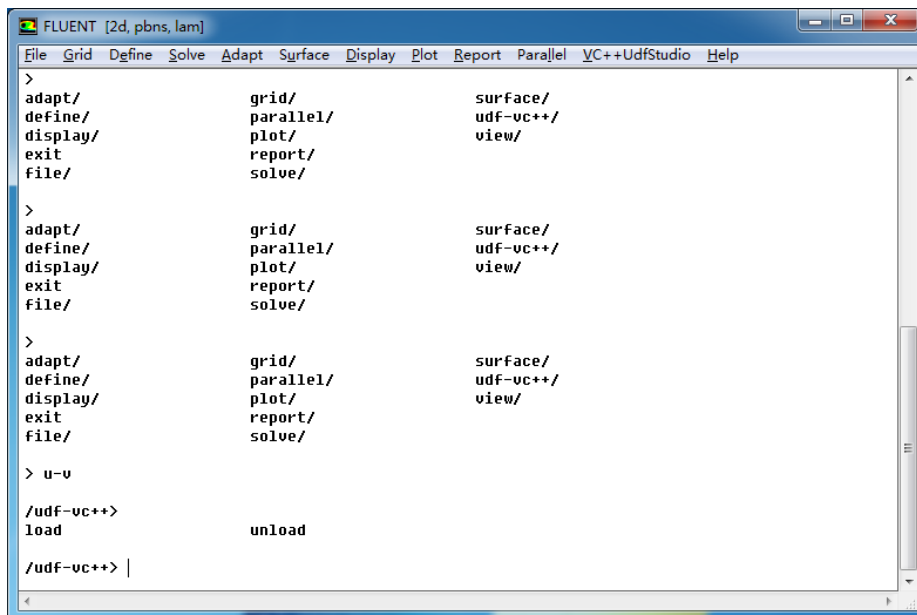
DEFINE_EXECUTE_ON_LOADING(load,libudf)
{
    SuperUdf_Initialize(AfxGetInstanceHandle());
}
```

10、小贴士

1. 强烈推荐安装“Visual assistant” (www.wholetomato.com) 插件。该插件具有很多扩展功能（例如代码自动补全，括号匹配，用户自定义关键词颜色等等）。



2. 利用 TUI 命令 `udf-vc++/load` 或 `udf-vc++/unload` 可以加载或卸载 Fluent 中的 VC++UdfStudio 菜单。

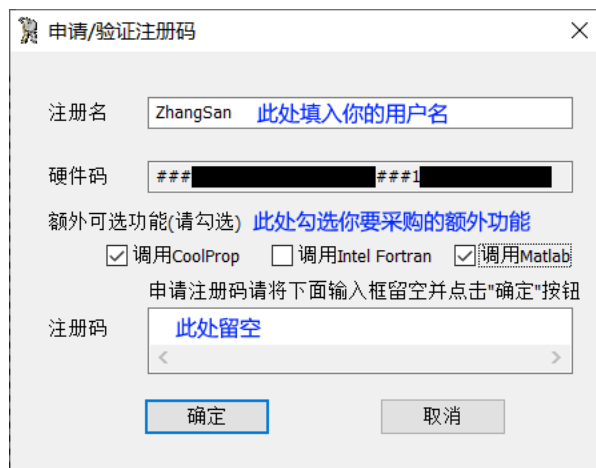


11、如何注册

1. 打开启动器并点击“Register”菜单。



2. 输入你的用户名，“Code”文本框留空，根据您的需要勾选额外需要采购的功能，“调用CoolProp”，“调用Intel Fortran”或“调用Matlab”功能，然后点击“OK”。您的用户名和硬件信息会被存放到文本文件“user.ini”。



3. 联系 vcUdfStudio@outlook.com (国际)或 vcUdfStudio@sohu.com (中国)并付款，确认成功后将“user.ini”文件作为附件发送，收到返回的带有注册号的 email 后，以管理员权限重新运行启动器。再次点击“Register”菜单，输入您的用户名和注册号，所有采购的功能即可解锁（注意：注册成功后，若未采购“调用CoolProp”，“调用Intel Fortran”或“调用Matlab”，对应功能将禁用，不再允许试用。想要恢复试用未采购的功能可以卸载后重新安装本插件，即可自动切换成试用版，从而开放该试用功能）。

